Croatian Operational Research Review
CRORR **10**(2019), 35-44

A Max-Plus algebra approach for generating a non-delay schedule

Tena Žužek[1,*], **Aljoša Peperko**[1,2], **Janez Kušar**[1]

[1] *Faculty of Mechanical Engineering, University of Ljubljana*
Aškerčeva 6, 1000 Ljubljana, Slovenia
E-mail: ⟨{tena.zuzek, aljosa.peperko, janez.kusar}@fs.uni-lj.si⟩

[2] *Institute of Mathematics, Physics and Mechanics*
Jadranska 19, 1000 Ljubljana, Slovenia
E-mail: ⟨aljosa.peperko@fmf.uni-lj.si⟩

Abstract. A Max-Plus algebra is one of the promising mathematical approaches that can be used for scheduling operations. It was already applied for the presentation of Johnson's algorithm and for solving cyclic jobshop problems, but it had not yet been applied for non-delay schedules. In this article, max-plus algebra is used to formally present the generation of a non-delay schedule for the first time. We present a simple algorithm for generating matrices of starting and finishing times of operations, using max-plus algebra formalism. We apply the LRPT (Longest Remaining Processing Time) rule as the priority rule, and the SPT (Shortest Processing Time) rule as the tie-breaking rule. The algorithm is applicable for any other pair of priority rules with a few minor adjustments.

Keywords: Max-Plus algebra, non-delay schedule, priority rules, project scheduling

Received: September 19, 2018; accepted: April 18, 2019; available online: July 4, 2019

DOI: 10.17535/crorr.2019.0004

1. Introduction

Scheduling operations is a very important task. An optimal schedule results in shorter lead times, reduced stocks, and lower production costs. An increase in the number of operations very rapidly increases the complexity of generating an optimal schedule. To generate an optimal schedule, all possibilities should be analyzed, but this is very time consuming and often unreasonable. That is why, in most cases, we settle for a sub-optimal solution.

Non-delay schedules are a subgroup of all possible schedules in which a free machine starts executing an operation as soon as it is schedulable [4, 7, 10, 12] or in other words, in a non-delay schedule no machine is kept idle at a time when it could begin processing some operation [1]. Non-delay schedules are relatively simple to generate and despite not being necessarily optimal, they mostly provide satisfactory results that are suboptimal [1]. A set of possible schedules can still be further narrowed by using priority rules.

When generating a schedule, various mathematical tools can be used in which priority rules are taken into consideration. One such tool is max-plus algebra. Max-plus algebra has already been used for the presentation of Johnson's algorithm [2]. For solving cyclic jobshop problems [6] and for studying production processes that run in cycles and involve several machines [3], but it has not yet been applied for non-delay schedules. One of the benefits of max-plus algebra is that it provides a setting in which classically nonlinear problems are described in a linear fashion. Moreover, it can be considered as an algebra for combinatorial optimization, and so the computational techniques from combinatorial optimization can be applied to such problems.

*Corresponding author.

The purpose of this research is to find whether max-plus algebra can be used to formally present the generation of non-delay schedules under the consideration of the LRPT priority rule and the SPT tie-breaking rule.

In the article that follows, the priority rules and the fundamental properties of a non-delay schedule and max-plus algebra will be firstly described. A developed algorithm for the generation of a non-delay schedule by means of max-plus algebra will be presented. A numerical example will be provided which will be compared with existing results in the literature.

2. Literature review

2.1. A non-delay schedule

The goal of scheduling operations is normally the generation of a schedule with the shortest lead time. There are usually many schedules that represent feasible solutions. Analyzing all schedules and searching for the best one is very time consuming, this is why we frequently settle for a solution that is close to the optimal one.

In a first step, a set of solutions is narrowed by considering only the semi-active schedules. The schedules are semi-active when none of the operations can be executed at an earlier time without changing the sequence of the operations or violating technological limitations and/or capacity limitations [1, 7]. The number of semi-active schedules increases dramatically with the number of machines and tasks. If there are m machines and n tasks, the number of semi-active schedules equals $(n!)^m$. If there are three machines and three tasks, the number of solutions equals 216, if there are four machines and four tasks, the number of solutions is as high as $3.32 \cdot 10^5$ [7].

In addition, let us have a look at a smaller subset of solutions, the so-called active schedules. A schedule is active when none of the operations can be executed at an earlier time without delaying the execution of any other operation or violating technological limitations and/or capacity limitations [7, 10]. An optimal schedule is always a part of the set of active schedules.

A still smaller subset is represented by non-delay schedules. They are defined as active schedules in which no machine is kept idle at a time when it could begin processing some operation. A set of non-delay schedules does not always include an optimal solution, yet the method of generating a non-delay schedule is relatively simple and the solutions are usually reasonable and sub-optimal. An algorithm for generating a non-delay schedule is described in detail in [1].

2.2. Priority rules

A number of non-delay schedules can further be reduced by using priority rules and tie-breaking rules. One final solution can thus be reached. There are many different priority rules that are based on information related to processing time, due date, the number of operations, arrival times, costs, setup times, slack, and so on. The majority of priority rules are summarized in [8] and [11]. We are only going to list some of the priority rules based on processing time. Note that the code names used for different rules may or may not coincide with other research:

- SPT rule gives priority to the operation with the shortest processing time
- LPT rule gives priority to the operation with the longest processing time
- SRPT rule gives priority to the task with the shortest remaining processing time (the remaining processing time can either include or exclude the processing time of the operation under consideration)

- LRPT rule gives priority to the task with the longest remaining processing time (the remaining processing time can either include or exclude the processing time of the operation under consideration)
- SSO rule prioritises task with the shortest subsequent operation
- LSO rule prioritises job with the longest subsequent operation

The selection of priority rules is not crucial for our research. We want to develop a max-plus algebra algorithm that can be used for generating non-delay schedule regardless of priority rule selected. We have chosen the LRPT rule as the priority rule, and the SPT rule as the tie-breaking rule. A decision for these two priority rules was made on the basis of the source [1] where a numerical example with the described two rules is presented. We will be able to make a comparison between the results of our research and those from the literature. Also, according to [1] the selected rules usually provide better solutions than other priority rules.

2.3. Max-Plus algebra

When scheduling operations, we should bear in mind that an operation of a certain task can only begin when a previous operation of the same task has been completed. Moreover, the machine on which an operation is performed should be available. The maximum of both such times determines the earliest possible start of an operation. If processing time is added to this earliest possible start, the time of completion of the operation is obtained.

The use of the operations of addition and maximum implies the idea that max-plus algebra can be a useful tool for generating schedules. This chapter will provide a definition of max-plus algebra and some of its main properties. Let us denote:

$$\varepsilon \stackrel{\text{def}}{=} -\infty \tag{1}$$

$$e \stackrel{\text{def}}{=} 0, \tag{2}$$

and let us define the set $\mathbb{R}_{max}$ by:

$$\mathbb{R}_{max} = \mathbb{R} \bigcup \{\varepsilon\}. \tag{3}$$

For the elements $a, b \in \mathbb{R}_{max}$, the operations $\oplus$ and $\otimes$ are defined:

$$a \oplus b \stackrel{\text{def}}{=} max(a, b) \tag{4}$$

$$a \otimes b \stackrel{\text{def}}{=} a + b. \tag{5}$$

Max-plus algebra can now be defined as the set $\mathbb{R}_{max}$ with the operations $\oplus$ and $\otimes$. It is denoted with $R_{max} = (\mathbb{R}_{max}, \oplus, \otimes, \varepsilon, e)$. Observe that the elements ε and e are the neutral elements for the operations $\oplus$ and $\otimes$ and that these operations play the role of addition and multiplication in this setting, respectively. For the operation $\oplus$ idempotency, commutativity, and associativity apply. For the operation $\otimes$ associativity and distributivity over $\oplus$ also hold [5, 12]. Algebraically, R_{max} is an idempotent semifield. By analogy to linear algebra, the operations $\oplus$ and $\otimes$ are extended to calculation with matrices and vectors.

Suppose the matrices $A, B \in \mathbb{R}_{max}^{n\times m}$ and the scalar $a \in \mathbb{R}_{max}$.
The following equations apply:

$$\left[A \oplus B\right]_{ij} = a_{ij} \oplus b_{ij} \tag{6}$$

$$\left[\alpha \otimes A\right]_{ij} = \alpha \otimes a_{ij} \ . \tag{7}$$

Let's also define the multiplication of matrices $A \in \mathbb{R}_{max}^{n\times l}$ and $B \in \mathbb{R}_{max}^{l\times m}$. The product of matrices $A \otimes B$ equals:

$$\left[A \otimes B\right]_{ik} = \oplus_{j=1}^{l} a_{ij} \otimes b_{jk} \tag{8}$$

3. A max-plus algebra algorithm for generating a non-delay schedule

To completely describe a schedule, we need the processing times of individual operations and the starting or finishing times of operations. To this purpose, we generate a matrix of processing times of operations T of dimensions $m \times n$, where m denotes the number of machines and n denotes the number of tasks. Element $\left[T\right]_{ij}$ equals the processing time of the task j on the machine i. Furthermore, we generate the matrix $N(k)$, also of dimensions $m \times n$. Element $\left[N(k)\right]_{ij}$ equals a completion time of the task j on the machine i. At the beginning ($k = 0$) all elements of the matrix equal the unit element e, and then in each stage, one element is updated. We generate the schedule in $m \times n$ stages. As already mentioned, in each stage of generating a non-delay schedule, the operation with the earliest possible start is scheduled. We denote an operation with index ij, where i denotes the machine on which the operation is executed, and j denotes the task to which the operation is assigned. The earliest possible start of an operation ij which can be scheduled in the $k-$th stage, is determined by finding the maximum value in column j and row i of the matrix $N(k-1)$. The maximum value in column j indicates when the task is available for execution, while the maximum value in row i indicates when the required machine is available. The maximum of both values indicates the earliest possible start of operation σ_{ij}. The set of all σ_{ij} in stage k is denoted with σ_k. The minimum value from set σ_k determines the operation which is scheduled in that stage.

If several operations share the same minimum value σ_{ij}, the priority rule is applied, and if there is another tie, an additional tie-breaking rule is applied. The priority rule gives priority to the task having the longest remaining processing time, while the tie-breaking rule gives priority to the task, the operation of which, that is being scheduled, has the shortest processing time. When it is determined which operation will be scheduled in stage k, the matrix $N(k)$ must be updated. The updated element of matrix $N(k)$, in the linear algebra language, equals the sum of the processing time of the selected operation and the adequate value σ_{ij}. We propose a systematic max-plus algebra approach for generating non-delay schedules under the consideration of the LRPT priority rule and the SPT tie-breaking rule. The procedure consists of six steps and works as follows.

A Max-Plus algebra approach for generating a non-delay schedule under the consideration of the LRPT priority rule and the SPT tie-breaking rule

Step 1:
Let $k = 1$. Generate the matrix of processing times of operations T of dimensions $(n \times m)$, where element $\left[T\right]_{ij}$ equals the processing time of the task j on the machine i
Generate the initial matrix of completion times of operations $N(0)$ of dimensions $(n \times m)$, where element $\left[N(0)\right]_{ij}$ equals e

Step 2:
Determine the operations ij that can be scheduled in stage k

Step 3:
For each operation from Step 2 determine the earliest possible start σ_{ij} by the equation:

$$\sigma_{ij} = \oplus_{l=1}^{M}\left[N(k-1)\right]_{lj} \oplus \oplus_{l=1}^{J}\left[N(k-1)\right]_{il} \tag{9}$$

Denote the set of all σ_{ij} in stage k with σ_k

Step 4:
The minimal value from the set σ_k determines the operation which is scheduled in stage k
a. If several operations share the same minimum value σ_{ij}, apply the LRPT priority rule
b. If there is another tie after applying the LRPT priority rule, apply the additional SPT tie-breaking rule

Step 5:
Update the matrix of completion times. The updated element $\left[N(k)\right]_{ij}$ equals the completion time of task j on the machine i by the equation:

$$\left[N(k)\right]_{ij} = \left[T\right]_{ij} \otimes \sigma_{ij} \tag{10}$$

Step 6:
Increment k by one. Return to Step 2 and continue in this manner until all operations have been scheduled (until $k \leq m \times n$).

Figure 1: *A Max-Plus algorithm*

3.1. Numerical example

Let us consider a case with $n = 4$ tasks and $m = 3$ machines. Table 1 shows the succession of execution of tasks on each machine, the processing times of operations are indicated in brackets. Under the consideration of the LRPT priority rule and the SPT tie-breaking rule, the example is solved in [1]. Figure 1 shows the Gantt chart of the obtained non-delay schedule.

J1	M1 (4)	M2 (3)	M3 (2)
J2	M2 (1)	M1 (4)	M3 (4)
J3	M3 (3)	M2 (2)	M1 (3)
J4	M2 (3)	M3 (3)	M1 (1)

Table 1: *The succession of execution of tasks on each machine and the processing times of operations for a numerical example with four tasks and three machines*

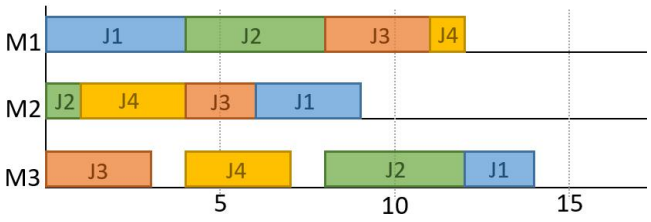

Figure 2: *The Gantt chart of the non-delay schedule from Table 1, obtained under the consideration of the default LRPT priority rule, and the SPT tie-breaking rule*

First, the matrices T and $N(0)$, both of dimensions 3×4, are generated:

$$T = \begin{pmatrix} 4 & 4 & 3 & 1 \\ 3 & 1 & 2 & 3 \\ 2 & 4 & 3 & 3 \end{pmatrix} \qquad N(0) = \begin{pmatrix} e & e & e & e \\ e & e & e & e \\ e & e & e & e \end{pmatrix} \tag{11}$$

In stage one the operations 11, 22, 33, 24 can be scheduled. In the matrix $N(0)$, the places of these operations are written in bold. The maximum value of an adequate row and column is determined for each of these four operations. In this case all values σ_{ij} equal e. The priority rule must be applied. The tasks 1 and 2 have the longest remaining processing time (9 time units). The tie-breaking rule must further be applied. The processing time of operation 22 equals 1, while the processing time of operation 11 equals 4. In stage one operation 22 is scheduled because it has a shorter processing time than operation 11. The updated element of matrix $N(1)$ equals:

$$\big[N(1)\big]_{22} = \big[T\big]_{22} \otimes \Big(\oplus_{l=1}^{3} \big[N(0)\big]_{l2} \oplus \oplus_{l=1}^{4} \big[N(0)\big]_{2l} \Big) = 1 \otimes (e \oplus e) = 1 \tag{12}$$

In matrix $N(1)$ the updated element is marked with a star, and the operations which can be scheduled in the next stage are again written in bold:

$$N(1) = \begin{pmatrix} \mathbf{e} & \mathbf{e} & e & e \\ e & 1^* & e & \mathbf{e} \\ e & e & \mathbf{e} & e \end{pmatrix} \tag{13}$$

In stage two the operations 11, 12, 33, 24 can be scheduled. Again, the maximum value of an adequate row and column is determined for each operation. The values σ_{ij} equal e, 1, e, 1, respectively. The priority rule determines execution of operation 11, the processing time equals 4 time units, the belonging σ_{11} equals e. The updated element and matrix $N(2)$ equal:

$$\big[N(2)\big]_{11} = \big[T\big]_{11} \otimes \Big(\oplus_{l=1}^{3} \big[N(1)\big]_{l1} \oplus \oplus_{l=1}^{4} \big[N(1)\big]_{1l} \Big) = 4 \otimes (e \oplus e) = 4 \tag{14}$$

$$N(2) = \begin{pmatrix} 4^* & \mathbf{e} & e & e \\ \mathbf{e} & 1 & e & \mathbf{e} \\ e & e & \mathbf{e} & e \end{pmatrix} \tag{15}$$

In stage three the operations 21, 12, 33, 24 can be scheduled. The values σ_{ij} equal 4, 4, e, 1, respectively. The minimum value $\sigma_3 = e$ belongs to operation 33, the processing time of which equals 3 time units. The updated element and matrix $N(3)$ equal:

$$\big[N(3)\big]_{33} = \big[T\big]_{33} \otimes \Big(\oplus_{l=1}^{3} \big[N(2)\big]_{l3} \oplus \oplus_{l=1}^{4} \big[N(2)\big]_{3l} \Big) = 3 \otimes (e \oplus e) = 3 \tag{16}$$

$$N(3) = \begin{pmatrix} 4 & \mathbf{e} & e & e \\ \mathbf{e} & 1 & \mathbf{e} & \mathbf{e} \\ e & e & 3^* & e \end{pmatrix} \tag{17}$$

The operations 21, 12, 23, 24 can be scheduled in stage four. The values σ_{ij} equal 4, 4, 3, 1, respectively. The minimum value $\sigma_4 = 1$ belongs to operation 24, the processing time of which equals 3 time units. The updated element and matrix $N(4)$ equal:

$$\big[N(4)\big]_{24} = \big[T\big]_{24} \otimes \Big(\oplus_{l=1}^{3} \big[N(3)\big]_{l4} \oplus \oplus_{l=1}^{4} \big[N(3)\big]_{2l} \Big) = 3 \otimes (e \oplus 1) = 4 \tag{18}$$

$$N(4) = \begin{pmatrix} 4 & \mathbf{e} & e & e \\ \mathbf{e} & 1 & \mathbf{e} & 4^* \\ e & e & 3 & \mathbf{e} \end{pmatrix} \tag{19}$$

Eight subsequent stages are carried out in this way. In each stage we find the operations that can be scheduled, their earliest possible starts, and if necessary the priority rule and the tie-breaking rule are applied. For the whole step-by-step procedure see the Appendix A. For each stage the updated matrix is given. The operations that can be processed in the next stage are denoted in bold, and the updated element is marked with a star. The final matrix $N(12)$ equals:

$$N(12) = \begin{pmatrix} 4 & 8 & 11 & 12 \\ 9 & 1 & 6 & 4 \\ 14 & 12 & 3 & 7 \end{pmatrix} \tag{20}$$

The elements of the final matrix indicate when each operation completes. The first row of the final matrix indicates when each task completes on the machine M1, the second row indicates when each task completes on the machine M2, and the third row indicates when each task completes on the machine M3. The maximum value in the matrix equals the total lead time. As evident, the lead time equals 14 time units.

If the matrix T is subtracted from the final matrix of end times $N(12)$, the matrix of times at which operations must start is obtained. We denote the matrix of starting times with S:

$$S = \begin{pmatrix} 0 & 4 & 8 & 11 \\ 6 & 0 & 4 & 1 \\ 12 & 8 & 0 & 4 \end{pmatrix} \tag{21}$$

We can see, that the results obtained with our max-plus algebra approach match those from the literature (see Gantt chart in Figure 1). In the source [1] we can also see that the sequence of scheduling operations is the same.

A numerical example with four tasks and three machines is considered as a relatively small-sized problem, but it allows a very clear visualisation of the proposed algorithm. Bigger-sized problems were also solved using Wolfram Mathematica. See the Appendix B for a numerical example with 15 tasks and 15 machines (a jobshop larger than 15×15 is considered to be complex [9]).

4. Conclusion

In the article a new max-plus algebra approach for a well-known non-delay heuristic is proposed. We applied the LRPT rule as the priority rule and the SPT rule as the tie-breaking rule. However, with minor changes, the proposed algorithm can be used for any other pair of priority rules. The solution of this max-plus algebra approach for generating non-delay schedules is the matrix $N(k)$, the elements of which represent the completion times of individual

operations and the maximum element equals the total lead time. If the matrix of processing times of operations T is subtracted from the matrix of completion times $N(k)$, the matrix of times at which operations must start is obtained. The main contribution of this article is a novel formalism for generating non-delay schedules. The greatest advantage of the developed procedure is in its simplicity. The final solution is presented in the form of a matrix, that enables one to quickly obtain the completion times of all the operations and the total lead time of the problem. The goal of further research is to upgrade the developed algorithm so that an improved schedule (a schedule with a shorter lead time) can be obtained. Ideally, the algorithm should allow generation of an optimal schedule in real time.

Acknowledgements

The authors acknowledge the financial support from the Slovenian Research Agency (research core fundings No. P1-0222, P2-0270 and J1-8133).

References

[1] Baker, K. R. (1974). *Introduction to Sequencing and Scheduling*. New York: John Wiley & Sons.
[2] Bouquard, J. L., Lenté, C. and Billaut, J. C. (2006). Application of an optimization problem in Max-Plus algebra to scheduling problems. *Discrete Applied Mathematics*, 154(15), 2064–2079. doi: 10.1016/j.dam.2005.04.011
[3] Butkovič P. (2010). *Max-linear Systems: Theory and Algorithms*. London: Springer Monographs in Mathematics. doi: 10.1007/978-1-84996-299-5
[4] Cohen, Y., Sadeh, A. and Zwikael, O. (2012). Finding the Shortest Non-Delay Schedule for a Resource-Constrained Project. *International Journal of Operations Research and Information Systems*, 3(4), 41–58. doi: 10.4018/joris.2012100103
[5] Heidergott, B., Olsder, G. J. and Woude, J. (2014). *Max Plus at Work. Modeling and Analysis of Synchronized Systems: A Course on Max-Plus Algebra and Its Applications*. New Jersey: Princeton Series in Applied Mathematics. doi: 10.1515/9781400865239
[6] Houssin, L. (2011). Cyclic jobshop problem and (max, plus) algebra. *IFAC Proceedings Volumes*, 44(1), 2717–2721. doi: 10.3182/20110828-6-it-1002.03095
[7] Murovec, B. (2002). Preprečevanje neizvedljivosti urnikov pri metahevrističnem razvrščanju proizvodnih procesov. Doctoral Dissertation. Ljubljana: Faculty of Electrical Engineering.
[8] Panwalkar, S. S. and Iskander, W. (1977). A survey of Scheduling Rules. *Operations Research*, 25(1), 45–61. doi: 10.1287/opre.25.1.45
[9] Sels, V., Vanhoucke, M. and Gheysen, N. (2012). A comparison of priority rules for the job shop scheduling problem under different flow time and tardiness-related objective functions. *International Journal of Production Research*, 50(15), 4255–4270. doi: 10.1080/00207543.2011.611539
[10] Sprecher, A., Kolisch, R. and Drexl, A. (1995). Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 80(1), 94–102. doi: 10.1016/0377-2217(93)e0294-8
[11] Tasič, T., Buchmeister, B. and Ačko, B. (2007). The Development of Advanced Methods for Scheduling Production Processes. *Journal of Mechanical Engineering*, 23(12), 844–857.
[12] Zwikael, O., Cohen, Y. and Sadeh, A. (2006). Non-delay scheduling as a managerial approach for managing projects. *International Journal of Project Management*, 24(4), 330—336. doi: 10.1016/j.ijproman.2005.11.002

Appendix A

In this appendix the stages 5 to 12 of generating a non-delay schedule of the numerical example from the Chapter 4 are presented. For each stage the updated matrix is given. The operations that can be processed in the next stage are denoted in bold and the updated element is marked with a star.

$$N(5) = \begin{pmatrix} 4 & 8^* & e & e \\ \mathbf{e} & 1 & \mathbf{e} & 4 \\ e & \mathbf{e} & 3 & \mathbf{e} \end{pmatrix} \quad N(6) = \begin{pmatrix} 4 & 8 & \mathbf{e} & e \\ \mathbf{e} & 1 & 6^* & 4 \\ e & \mathbf{e} & 3 & \mathbf{e} \end{pmatrix} \quad N(7) = \begin{pmatrix} 4 & 8 & \mathbf{e} & \mathbf{e} \\ \mathbf{e} & 1 & 6 & 4 \\ e & \mathbf{e} & 3 & 7^* \end{pmatrix}$$

$$N(8) = \begin{pmatrix} 4 & 8 & \mathbf{e} & \mathbf{e} \\ 9^* & 1 & 6 & 4 \\ \mathbf{e} & \mathbf{e} & 3 & 7 \end{pmatrix} \quad N(9) = \begin{pmatrix} 4 & 8 & \mathbf{e} & \mathbf{e} \\ 9 & 1 & 6 & 4 \\ \mathbf{e} & 12^* & 3 & 7 \end{pmatrix} \quad N(10) = \begin{pmatrix} 4 & 8 & 11^* & \mathbf{e} \\ 9 & 1 & 6 & 4 \\ \mathbf{e} & 12 & 3 & 7 \end{pmatrix}$$

$$N(11) = \begin{pmatrix} 4 & 8 & 11 & 12^* \\ 9 & 1 & 6 & 4 \\ \mathbf{e} & 12 & 3 & 7 \end{pmatrix} N(12) = \begin{pmatrix} 4 & 8 & 11 & 12 \\ 9 & 1 & 6 & 4 \\ 14^* & 12 & 3 & 7 \end{pmatrix}$$

Appendix B

Here we provide a numerical example with 15 tasks and 15 machines. A problem of this size is considered to be complex. Table shows the succession of execution of tasks on each machine, the processing times of operations are indicated in brackets.We solved the problem using Wolfram Mathematica. We used the proposed algorithm for generating non-delay schedules. If there was another tie even after applying the tie-breaking rule, the algorithm schedules the operation of the task with the lowest index (operation of task J1 has higher priority than operation of task J2 and so on).

J1	J2	J3	J4	J5	J6	J7	J8
M1 (1)	M3 (3)	M8 (1)	M2 (3)	M1 (7)	M15 (6)	M14 (3)	M8 (1)
M2 (3)	M1 (3)	M6 (1)	M15 (4)	M2 (4)	M14 (4)	M12 (1)	M1 (1)
M3 (8)	M2 (1)	M10 (1)	M1 (6)	M9 (1)	M13 (1)	M10 (6)	M15 (1)
M4 (1)	M12 (1)	M7 (2)	M14 (4)	M10 (1)	M12 (1)	M8 (1)	M14 (3)
M5 (2)	M13 (4)	M3 (8)	M5 (6)	M14 (2)	M11 (6)	M15 (2)	M3 (1)
M6 (1)	M4 (3)	M4 (1)	M8 (3)	M15 (6)	M10 (7)	M6 (1)	M2 (1)
M7 (2)	M6 (1)	M5 (1)	M9 (1)	M13 (2)	M9 (8)	M4 (1)	M10 (5)
M8 (3)	M5 (2)	M14 (3)	M13 (2)	M12 (3)	M8 (2)	M13 (1)	M6 (7)
M9 (4)	M14 (2)	M15 (3)	M3 (7)	M11 (1)	M7 (3)	M2 (1)	M11 (4)
M10 (1)	M7 (4)	M13 (2)	M12 (1)	M4 (3)	M6 (2)	M1 (9)	M7 (6)
M11(1)	M15 (2)	M11 (4)	M11 (5)	M5 (8)	M5 (1)	M3 (5)	M12 (1)
M12(5)	M8 (1)	M9 (1)	M10 (7)	M7 (6)	M4 (2)	M5 (2)	M9 (1)
M13(1)	M10 (3)	M1 (1)	M7 (1)	M8 (9)	M3 (1)	M7 (3)	M4 (11)
M14(2)	M11 (4)	M12 (1)	M4 (7)	M6 (1)	M2 (5)	M9 (4)	M5 (10)
M15(1)	M9 (2)	M2 (2)	M6 (5)	M3 (1)	M1 (8)	M11(8)	M13 (2)

J9	J10	J11	J12	J13	J14	J15
M5 (9)	M1 (2)	M6 (1)	M6 (3)	M5 (1)	M6 (1)	M15 (1)
M12 (7)	M2 (3)	M7 (4)	M10 (3)	M3 (2)	M4 (6)	M14 (3)
M1 (1)	M3 (1)	M10 (1)	M11 (5)	M1 (4)	M1 (2)	M13 (2)
M15 (1)	M7 (4)	M1 (3)	M12 (1)	M14 (5)	M2 (2)	M12 (2)
M7 (5)	M8 (2)	M2 (4)	M5 (1)	M9 (4)	M3 (5)	M1 (1)
M10 (3)	M9 (1)	M14 (2)	M9 (3)	M11 (5)	M5 (7)	M2 (3)
M4 (12)	M12 (2)	M13 (1)	M8 (3)	M13 (2)	M7 (2)	M3 (4)
M2 (2)	M13 (8)	M11 (4)	M7 (1)	M4 (3)	M8 (2)	M5 (1)
M14 (1)	M14 (3)	M15 (1)	M4 (2)	M7 (3)	M9 (5)	M4 (5)
M9 (2)	M10 (2)	M3 (2)	M3 (2)	M2 (1)	M10 (1)	M6 (4)
M11 (6)	M6 (3)	M9 (2)	M13 (2)	M15 (3)	M15 (1)	M11 (1)
M13 (8)	M4 (1)	M4 (6)	M14 (1)	M12 (3)	M14 (4)	M10 (1)
M6 (8)	M5 (2)	M12 (3)	M15 (2)	M10 (2)	M13 (4)	M9 (6)
M3 (1)	M11 (4)	M5 (5)	M2 (5)	M8 (3)	M12 (4)	M8 (3)
M8 (1)	M15 (6)	M8 (2)	M1 (1)	M6 (8)	M11 (6)	M7 (1)

The final matrix of completion times equals:

$$N = \begin{pmatrix} 24 & 30 & 86 & 16 & 7 & 85 & 39 & 8 & 17 & 23 & 27 & 92 & 21 & 10 & 40 \\ 29 & 34 & 93 & 3 & 11 & 77 & 18 & 20 & 42 & 26 & 33 & 91 & 68 & 13 & 45 \\ 37 & 3 & 27 & 51 & 78 & 72 & 44 & 19 & 73 & 28 & 62 & 81 & 12 & 18 & 55 \\ 41 & 44 & 28 & 87 & 47 & 71 & 16 & 61 & 40 & 80 & 77 & 79 & 64 & 7 & 69 \\ 43 & 48 & 33 & 31 & 56 & 61 & 46 & 71 & 9 & 82 & 87 & 32 & 10 & 25 & 57 \\ 44 & 45 & 6 & 92 & 77 & 60 & 15 & 32 & 75 & 58 & 2 & 5 & 87 & 1 & 76 \\ 46 & 54 & 18 & 72 & 63 & 57 & 49 & 42 & 23 & 32 & 6 & 64 & 69 & 27 & 89 \\ 49 & 57 & 3 & 34 & 72 & 51 & 11 & 1 & 74 & 36 & 90 & 56 & 79 & 29 & 88 \\ 57 & 87 & 78 & 38 & 12 & 47 & 61 & 50 & 49 & 39 & 64 & 53 & 32 & 37 & 85 \\ 58 & 61 & 16 & 71 & 15 & 35 & 10 & 25 & 28 & 55 & 11 & 14 & 76 & 38 & 79 \\ 65 & 82 & 77 & 64 & 42 & 25 & 73 & 36 & 55 & 86 & 59 & 19 & 41 & 92 & 78 \\ 70 & 35 & 87 & 52 & 34 & 17 & 4 & 43 & 16 & 41 & 80 & 22 & 74 & 60 & 21 \\ 71 & 39 & 66 & 41 & 31 & 11 & 17 & 73 & 64 & 49 & 52 & 83 & 51 & 56 & 19 \\ 73 & 50 & 38 & 21 & 23 & 10 & 3 & 14 & 44 & 53 & 35 & 84 & 28 & 43 & 17 \\ 74 & 56 & 42 & 10 & 29 & 6 & 14 & 11 & 18 & 92 & 60 & 86 & 71 & 39 & 12 \end{pmatrix}$$