do not overlap by (2), so the assignment of p_j is feasible. If job p_j is scheduled in Step 5, it is easy to see that the time slots assigned to the two parts of p_j do not overlap. Clearly, there is a machine to be removed from S_{NFVM} in Step 4 or 5. □

Lemma 2. For any job p_j, we have $p_j \le W_e = \int_{l_e}^{w_Q} v_e(t)\,dt$, where $e = \min_{V_i \in S_{NFVM}}\{i\}$.

Proof. If $e>1$, i.e., the machines $V_1, V_2, ..., V_{e-1}$ have been removed from S_{NFVM}. According to Lemma 3.1(ii), once a job is processed in accordance with Step 4 or Step 5, it involves two machines, and the machine with the larger subscript becomes a full machine and is removed. Therefore, together with the definition of w_Q, the fact that the first $e-1$ machines are removed indicates that the first $e-1$ largest jobs, i.e., $p_1^{\max}$, $p_2^{\max}$, ..., $p_{e-1}^{\max}$, are exactly processed on the first $e-1$ virtual machines before scheduling p_j. Moreover, if $l_e = r_e$, i.e., there is no job on V_e before scheduling p_j, we must have $p_j \le W_e$, since the first e largest jobs can be processed on the first e virtual machines by the definitoin of w_Q. So, we consider the case where $l_e > r_e$, i.e., there are some jobs (or parts of the jobs) on V_e before scheduling p_j.

If there is only one virtual machine V_e in S_{NFVM}, i.e., all the other virtual machines have been removed with loads of w_Q, we must have $p_j \le W_e$, as all the jobs can be processed on the m virtual machines before time w_Q.

Now, we focus on the case where there are at least two virtual machines in S_{NFVM}. It is easy to derive that all the jobs (or parts) on V_e must be scheduled in Step 4 or Step 5. Otherwise, if a job (or parts) on V_e is scheduled in Step 3, then V_e must be the only one in S_{NFVM}, a contradiction. By the rule of Step 4 or Step 5, we obtain that V_{e+1} must be removed. Let V_{k+2} be the current adjacent virtual machine of V_e in S_{NFVM}, $e \le k \le m-2$, i.e., all the $k-e+1$ virtual machines between V_e and V_{k+2} have been removed from S_{NFVM}. Because $V_{k+2} \in S_{NFVM}$, there is no job processed on both V_{k+2} and any one of the first $k+1$ virtual machines. Moreover, by Steps 4 and 5, and Observation 1, we conclude that there are exactly k jobs, i.e., $p_{j1}, p_{j2}, ..., p_{jk}$, on the first $k+1$ machines before scheduling p_j. By the definition of w_Q, we have

$$\sum_{i=1}^{k+1}\int_{r_i}^{w_Q} v_i(t)dt \ge \sum_{i=1}^{k+1}\int_{r_i}^{T_{k+1}} v_i(t)dt = \sum_{i=1}^{k+1} p_i^{\max} \ge \sum_{i=1}^{k} p_{ji} + p_j ,$$

which yields $p_j \le \int_{l_e}^{w_Q} v_e(t)dt = W_e$, given that $\sum_{i=1}^{k} p_{ji} > \sum_{i=1}^{e-1}\int_{r_i}^{w_Q} v_i(t)dt + \sum_{i=e+1}^{k+1}\int_{r_i}^{w_Q} v_i(t)dt$. □

By Observation 1, and Lemmas 1 and 2, we obtain the following result.

Theorem 1. Algorithm A generates an optimal schedule.

We now consider the number of preemptions. There are two types of preemptions in the algorithm. One is produced by partitioning a job into two parts and scheduling them on two different virtual machines like in Steps 4 and 5. The other is produced by the different speeds on the virtual machines. Clearly, there are at most $m-1$ peemptions for the first type of preemptions.

For the second type, it suffices to calculate the number of speeds of each virtual machines. Initially, the number of speeds of virtual V_i is at most $m+1-i$ after converting all the m uniform machines to m virtual machines. It follows that there are at most $m-i$ preemptions on V_i, so the total number of preemptions is at most $\sum_{i=1}^{m}(m-i) = \frac{m^2-m}{2}$. Note that the speeds on the two virtual machines V_h and V_k are unchanged in Step 4, so we have no new preemptions. However, when a job is processed in Step 5, one more preemption is introduced because the time slot $[l_k, T]$ of the new virtual machine V_h is from the virtual machine V_k. This produces a maximum of $m-1$ preemptions. Hence, the total number of preemptions is at most

$$\frac{m^2-m}{2} + 2(m-1) = \frac{m^2+3m}{2} - 2 .$$

To illustrate Algorithm A, we present an example as follows:

Example. There are four jobs and three ma-

chines as shown in Table 1.

Table 1 An example

j	1	2	3	4
p_j	12	11	2	4
r_j	1	4	5	
s_j	2	3	4	

It is easy to find that $w_Q = 7$ for the above instance. Fig. 3 shows the virtual machines and the schedule of the jobs. The first job p_1 is assigned to V_1 and V_2 in Step 4, and V_2 is removed from S_{NFVM}. Job p_2 is assigned to V_1 and V_3 in Step 5, and the reminding time slots of V_1 and V_3 are spliced together into one time slot of the new virtual machine V_1. Finally, p_3 and p_4 are scheduled on V_1 in Step 3.

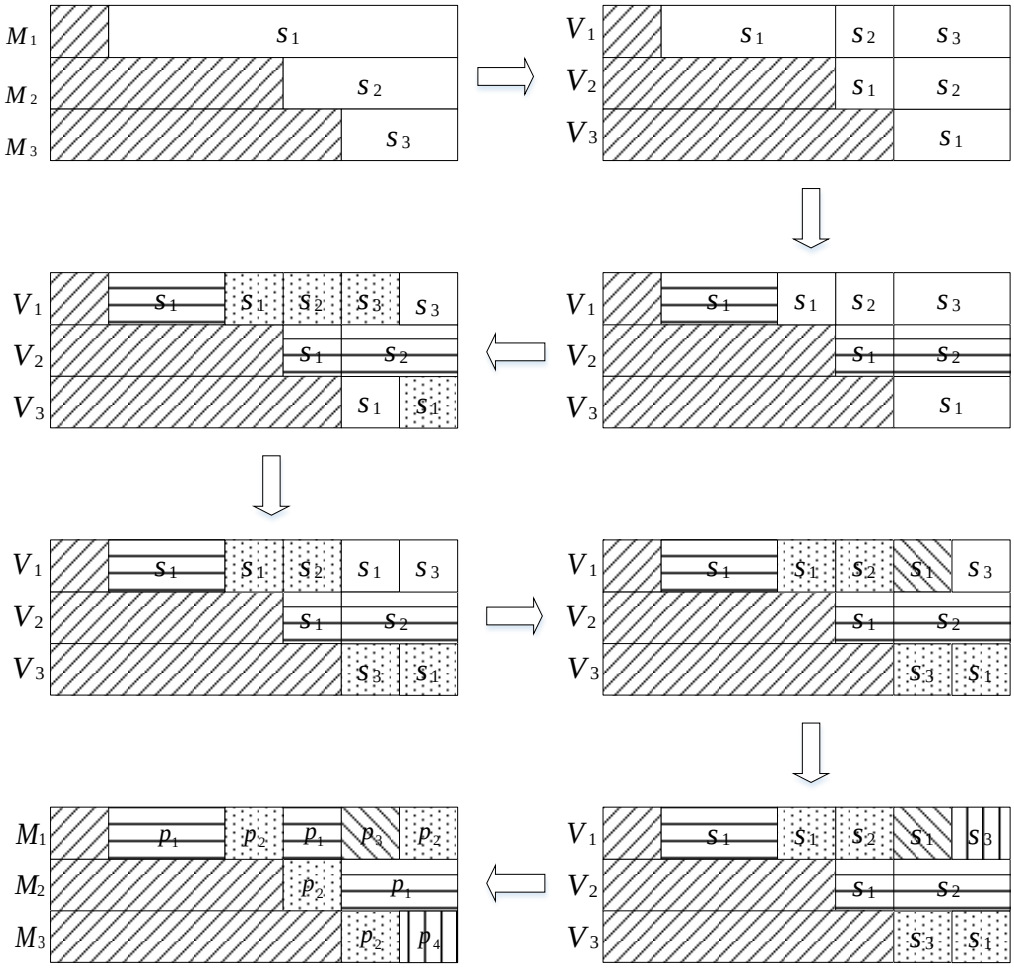

Fig. 3. The scheduling process of the example.

4 Conclusions

In this paper, we studied preemptive scheduling on parallel uniform machines with non-simultaneously available time. Our objective was to minimize the makespan. We provided a lower bound on the optimal makespan by transforming the m uniform machines into m virtual machines, ensuring that a machine with an earlier available time has a greater speed at any time. We provided a solution algorithm for the problem. For future research, it is worth considering the online version of the problem for different parallel-machine settings, including identical machines and uniform machines.

Contributors

Hao ZHOU designed the research. Li-ping CAO and Qi WEI designed and analyzed the algorithm. Hao ZHOU and Li-ping CAO drafted the manuscript. Zhen-yu SHU helped organize the manuscript. Yi-wei JIANG revised and finalized

the paper.

Compliance with ethics guidelines

Hao ZHOU, Li-ping CAO, Qi WEI, Zhen-yu SHU and Yi-wei JIANG declare that they have no conflict of interest.

References

Błażewicz J, Dell'Olmo P, Drozdowski M, et al., 2003. Scheduling multiprocessor tasks on parallel processors with limited availability. Eur J Oper Res, 149(2):377-389. https://doi.org/10.1016/S0377-2217(02)00760-9

Chang SY, Hwang HC, 1999. The worst-case analysis of the MULTIFIT algorithm for scheduling nonsimultaneous parallel machines. Discrete Appl Math, 92(2-3):135-147. https://doi.org/10.1016/S0166-218X(99)00049-9

Gonzalez T, Sahni S, 1978. Preemptive scheduling of uniform processor systems. J ACM, 25(1):92-101. https://doi.org/10.1145/322047.322055

Grigoriu L, Friesen DK, 2017. Approximation for scheduling on uniform nonsimultaneous parallel machines. J Sched, 20(6):593-600. https://doi.org/10.1007/s10951-016-0501-1

He Y, 1998. Parametric LPT-bound on parallel machine scheduling with non-simultaneous available time. Asia-Pac J Oper Res, 15(1):29-36.

He Y, 2000. Uniform machine scheduling with machine available constraints. Acta Math Appl Sin, 16(2):122-129. https://doi.org/10.1007/BF02677672

Horvath EC, Lam S, Sethi R, 1977. A level algorithm for preemptive scheduling. J ACM, 24(1):32-43. https://doi.org/10.1145/321992.321995

Huo YM, 2019. Parallel machine makespan minimization subject to machine availability and total completion time constraints. J Sched, 22(4):433-447. https://doi.org/10.1007/s10951-017-0551-z

Hwang HC, Lim K, 2014. Exact performance of MULTIFIT for nonsimultaneous machines. Discrete Appl Math, 167:172-187. https://doi.org/10.1016/j.dam.2013.10.022

Kaabi J, Harrath Y, 2019. Scheduling on uniform parallel machines with periodic unavailability constraints. Int J Prod Res, 57(1):216-227. https://doi.org/10.1080/00207543.2018.1471242

Kurt A, Çetinkaya FC, 2024. Unrelated parallel machine scheduling under machine availability and eligibility constraints to minimize the makespan of non-resumable jobs. Int J Ind Eng Manage, 15(1):18-33. https://doi.org/10.24867/IJIEM-2024-1-345

Lee CY, 1991. Parallel machines scheduling with nonsimultaneous machine available time. Discrete Appl Math, 30(1):53-61. https://doi.org/10.1016/0166-218X(91)90013-M

Lee CY, Lei L, Pinedo M, 1997. Current trends in deterministic scheduling. Ann Oper Res, 70:1-41. https://doi.org/10.1023/A:1018909801944

Lee CY, He Y, Tang GC, 2000. A note on "parallel machine scheduling with non-simultaneous machine available time". Discrete Appl Math, 100(1-2):133-135. https://doi.org/10.1016/S0166-218X(99)00201-2

Liu JWS, Yang AT, 1974. Optimal scheduling of independent tasks on heterogeneous computing systems. Proc 1974 Annual Conf, p.38-45. https://doi.org/10.1145/800182.810377

Mahjoub A, Kaabi J, Harrath Y, 2021. Absolute bounds of list algorithms for parallel machines scheduling with unavailability periods. Int Trans Oper Res, 28(3):1594-1610. https://doi.org/10.1111/itor.12589

McNaughton R, 1959. Scheduling with deadlines and loss functions. Manage Sci, 6(1):1-140. https://doi.org/10.1287/mnsc.6.1.1

Muntz RR, Coffman EG, 1970. Preemptive scheduling of real-time tasks on multiprocessor systems. J ACM, 17(2):324-338. https://doi.org/10.1145/321574.321586

Santoro MC, Junqueira L, 2023. Unrelated parallel machine scheduling models with machine availability and eligibility constraints. Comput Ind Eng, 179:109219. https://doi.org/10.1016/j.cie.2023.109219

Schmidt G, 1984. Scheduling on semi-identical processors. Z für Oper Res, 28(5):153-162. https://doi.org/10.1007/BF01920917

Shen LX, Wang D, Wang XY, 2013. Parallel-machine scheduling with non-simultaneous machine available time. Appl Math Modell, 37(7):5227-5232. https://doi.org/10.1016/j.apm.2012.09.053

Wang XY, Zhou ZL, Ji P, et al., 2014. Parallel machines scheduling with simple linear job deterioration and non-simultaneous machine available times. Comput Ind Eng, 74:88-91. https://doi.org/10.1016/j.cie.2014.05.003

Wang XY, Hu XP, Liu WG, 2015. Scheduling with deteriorating jobs and non-simultaneous machine available times. Asia-Pac J Oper Res, 32(6):1550049. https://doi.org/10.1142/S0217595915500499