Approximation Algorithms for Multitasking Scheduling Problems

FEIFENG ZHENG1, ZHAOJIE WANG1, MING LIU2, (Senior Member, IEEE), AND CHENGBIN CHU3

1Glorious Sun School of Business and Management, Donghua University, Shanghai 200051, China
2School of Economics and Management, Tongji University, Shanghai 200092, China
3School of Economics and Management, Fuzhou University, Fuzhou 350108, China
Corresponding author: Chengbin Chu (w1856@fzu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 71771048, Grant 71832001, Grant 71531011, and Grant 71571134, and in part by the Fundamental Research Funds for the Central Universities under Grant 2232018H-07.

ABSTRACT In this work, we incorporate human factors and real-life operations into newly proposed multitasking scheduling problems with periodic shift activities. It is motivated by personnel resource scheduling with periodic work shifts under the requirement of providing continuous service to customers. We model the problem as two identical parallel machine scheduling with complementary non-available time periods, and consider two models with the objectives of the makespan, i.e. the maximum completion time and respectively the total completion time. We then prove that the Greedy algorithm and SPT rule are of asymptotic and parametric approximation ratios for the two models, respectively.

INDEX TERMS Multitasking, scheduling, periodic shift, approximation algorithm.

I. INTRODUCTION

In many scheduling applications, human resources are now becoming one of the critical factors or even the main issues in person-involved scheduling scenarios. Different from machines, people’s working efficiency and quality may decrease severely after many hours of work, and consequently they need work shifts. Plenty of papers have been searching for effective shift scheduling strategies to alleviate the above negative effect. In this work, we are inspired by the observation that several persons or work groups may own one shared machine resource. Some of such applications are long distance transportation by truck (drivers can be regarded as parallel machines, and the total mileage can be divided into n sections, which can be regarded as n jobs), medical teams for outpatient and emergencies (medical teams for outpatient and emergencies can be regarded as parallel machines, and patients can be regarded as jobs), etc. Two drivers may share one truck. When one driver is tired, he will rest (can be regarded as machine maintenance) and the other driver will drive. In this way the truck runs all day and all night, while the two drivers can recover energy by having enough rest in their respective nonworking time.

Motivated by the above applications, we incorporate multitasking into parallel machine problem with periodic available time, and design approximation algorithms for different objectives. Our main contributions are proving that the Greedy algorithm and SPT rule are of asymptotic and parametric approximation ratios for the two models, respectively. This is of great significance to the multitasking scheduling problem when the number of jobs is large enough.

II. LITERATURE REVIEW

Scheduling is regarded as an effective tool to help make production plans (Wu and Che, 2020). Recently there is a trend to study machine scheduling under multitasking. Multitasking is a concept originating in computer systems such that multiple processing tasks share a single processing resource such as CPU. Related work refers to Chan et al. (2008), Rogers et al. (2014), Zhang et al. (2014), Mostafa and Kusakabe (2016), and etc. In the application of operations management, many works investigate the characteristics and influences of widely existed multitasking phenomena (Shakeri and Logendran, 2007; Watson and Strayer, 2010; Coviello et al., (2014). Gaglioppa et al. (2008) study multitasking and multistage production plan and scheduling methods in manufacturing industries, in which a manufacturing system outputs several products in one process.
Constraint-based functions are designed to solve the proposed mixed integer programming (MIP). Hall et al. (2016) address two multitasking models featured with alternate period processing and shared processing, examine the complexity of the resulting scheduling problems and propose fast algorithms. Zhu et al. (2017a) study multitasking scheduling problems with a rate-modifying activity, considering human operators, and conduct four classical objectives to illustrate the theorems and algorithms. Zhu et al. (2017b) generalize their work to the setting with multiply rate-modifying activities. Liu et al. (2017) investigate a joint multitasking scheduling and common due date assignment problem on a single machine, and devise a MIP model with the objective to minimize the total penalties. Xiong et al. (2019) consider the multitasking scheduling problem on unrelated parallel machines to minimize the total weighted completion time. For solving this problem, they propose an exact branch and price algorithm.

For the scheduling problem with periodic available time constraint, Hsu et al. (2011) study m identical parallel-machine scheduling with rate-modifying activities to minimize the total completion time, and improve the polynomial time of previous work. Vallada and Ruiz (2011) present a local-search-included genetic algorithm for the unrelated parallel machine scheduling problem in which machine and job sequence dependent setup times are considered. They develop a benchmark of small and large instances to carry out the computational experiments. Shen et al. (2013) consider the problem of parallel-machine scheduling where jobs are available at time zero but the machines may not be available simultaneously at the time. They present polynomial time algorithms for two different models with either completion time or waiting time related objectives. Wang et al. (2014) deal with parallel machine scheduling of deteriorating jobs where the actual processing of a job is a linear function of its starting time, aiming at minimizing the logarithm of makespan. Ji et al. (2019) consider the problem of parallel-machine scheduling with machine-dependent slack (SLK) due-window assignment in the multitasking environment. For other related studies, please refer to Wardono and Fathi (2004), James and Almada-Lobo (2011), Hierons et al. (2016), and Jiang and Tan (2016).

In this paper, we study the multitasking scheduling problem with alternate working periods. Different people take use of one shared resource in alternate working periods, which can be described as identical parallel machines with alternate non-available time. At any time point either machine is being available for processing jobs, implying that the shared resource is continuously available all the time. Moreover, the resource may stop processing and hold on an unfinished job while start to process another job, and later on it shifts back to resume the processing of the previously interrupted one. To be explained in the two parallel machines environment, the processing of any job may be preempted (the job being processed is interrupted before completion) by a non-availability period T on one machine, say machine 1, while another job is started at the same time point on machine 2 due to the start of an available time period on that machine. Machine 1 resumes its job processing after the end of time period T. That is, by the concept of multitasking, more than one task is in the status of being processed via some resource at a time. For example, one job is in process while another job is on hold.

For the considered multitasking scheduling problem with alternate working periods, our contributions mainly include: (i) embedding multitasking into parallel machine scheduling problem with alternate periodic available time, (ii) constructing two models with widely studied respective objectives, i.e., the makespan and the total completion time, (iii) devising an approximate algorithm for each model. It is assumed in both models that the processing time of any job is a positive integer (or is scaled to be integer). We prove that Greedy algorithm is asymptotically max\{2(1 + \frac{a}{b}), 2(1 + \frac{b}{a})\}-approximation for the makespan objective, while SPT (shortest processing time) rule is asymptotically (1 + max\{\frac{a}{b}, \frac{b}{a}\})-approximation for the other objective.

The reminder of this paper is organized as follows. A detailed problem description is given in Section 2. Sections 3 and 4 discuss the objective of the makespan and total completion time, respectively. Section 5 draw the conclusion and some further research directions.

### III. PROBLEM DESCRIPTION

There are \( n \) jobs to be processed on two identical parallel machines 1 and 2. Each job \( j \) is of integer processing time \( p_{ij} \in \mathbb{Z}^+ \) (\( \mathbb{Z}^+ \) is a set of positive integers). Machine 1 is with periodic maintenance activities during time intervals \( [i(a + b), i(a + b) + a] \) where \( i = 0, 1, 2, \ldots \). That is, the machine is unavailable for processing jobs within the time intervals. Similarly, machine 2 is unavailable within time intervals \( [i(a + b) + a, (i + 1)(a + b)] \) where \( i = 0, 1, 2, \ldots \). Machines 1 and 2 have periodic available time intervals with length equal to \( a \) and \( b \), respectively. On each machine, jobs can be started in any available time interval, interrupted by some shift activity (each job can be preempted), and then resume its processing at the end of the shift activity. To the best of our knowledge, when combine with available time intervals on both machines, it becomes on continuous available machine with alternate intervals \( a \) and \( b \).

### IV. THE MODEL OF MAKESPAN

In this section, we consider the objective of makespan. We focus on the approximation performance of Greedy algorithm with the time complexity of \( O(n) \) (the pseudocode as shown in Algorithm 1) which assigns jobs one by one to either machine with currently smallest completion time, and processes jobs without any idleness (except for unavailable time, i.e. the machine is in maintenance activities) in between on both machines.

**Theorem 1:** For the model to minimize the makespan, Greedy algorithm is asymptotically max\{2(1 + \frac{a}{b}), 2(1 + \frac{b}{a})\}-approximation.
Algorithm 1 Greedy Algorithm 1

Require: n, p, Γ (a job input instance)
1: for j = 1 : n do
2: Calculate the completion time of job j on machine 1 and machine 2, i.e. c1j and c2j
3: if c1j < c2j then
4: Job j is assigned to machine 1
5: else
6: Job j is assigned to machine 2
7: end if
8: end for

Proof: Given a job input instance Γ, let σ = (J1, J2, ..., Jn) be the processing schedule by Greedy algorithm where jobs are sequenced in the non-decreasing order of their start times, and σ = σ1 ∪ σ2 where sub-schedules σ1 = (J[1,1], J[1,2], ..., J[1,n1]) and σ2 = (J[2,1], J[2,2], ..., J[2,n2]) contain jobs to be processed on machines 1 and 2 respectively, n1 + n2 = n.

Define by t1 and respectively t2 the time at which machine 1 and machine 2 complete their last jobs in Γ \ Jn (excluding Jn). Define by S1 = ∑j∈σ1 \ Jn pJ + S2 = ∑j∈σ2 \ Jn pJ the total processing time of jobs, excluding job Jn, on the two machines respectively. We have S1 + S2 = ∑j∈pJ − pJ, i.e.,

\[ t1 - \lfloor \frac{t1}{a+b} \rfloor a + t2 - \lfloor \frac{t2}{a+b} \rfloor b = \sum_{j \in \sigma} p_J - p_J \]

where \([\ast] \), \([\ast] \) indicate rounding up and down for \(*\) respectively.

In an optimal schedule, the makespan

\[ C^* \geq \left\{ \min \{t1, t2\}, p_n, \frac{1}{2} \sum_{J \in \sigma} p_J \right\} \]

We consider the following two cases by whether \(t1 \leq t2\).

Case 1. \(t1 \leq t2\). By Equation (1) and \(0 < p_n\),

\[ t1 + t2 = \lfloor \frac{t1}{a+b} \rfloor a + \lfloor \frac{t2}{a+b} \rfloor b + \sum_{J \in \sigma} p_J - p_J \]

\[ < (\frac{t1}{a+b} + 1)a + \frac{t2}{a+b} b + \sum_{J \in \sigma} p_J \]

By the above inequality, \(t2 < \frac{a+ \sum_{J \in \sigma} p_J}{a+b} < (a+b) + \frac{a+b}{a} \sum_{J \in \sigma} p_J\). In the Greedy schedule, the last job Jn is assigned to machine 1 since \(t1 \leq t2\).

\[ C_{greedy} \leq \max\{t1 + \lfloor \frac{p_n}{a+b} \rfloor a + p_n, t2\} \]

\[ \leq \max\{t1 + (\frac{p_n}{a+b} + 1)a + p_n, t2\} \]

\[ \leq \max\{t1 + (\frac{a+b}{a+b} + 1)p_n + a, a+b + \frac{a+b}{a} \sum_{J \in \sigma} p_J\} \]

(4)

For the makespan of an optimal schedule, by the case condition \(t1 \leq t2\) and Inequality (2), we have \( C^* \geq \{t1, p_n, \frac{1}{2} \sum_{J \in \sigma} p_J\}\).

We bound the ratio of \( \frac{C_{greedy}}{C^*} \) as follows.

\[ \frac{C_{greedy}}{C^*} \leq \max\{1 + (\frac{a}{a+b} + 1) + \frac{a+b}{C^*} + \frac{a+b}{a} \sum_{J \in \sigma} p_J\} \]

\[ \leq \max\{1 + (\frac{a}{a+b} + 1) + \frac{a+b}{C^*} + \frac{2(a+b)}{a}\}. \]

Thus, \( \lim_{C \to \infty} \frac{C_{greedy}}{C^*} = \max\{2 + \frac{a}{a+b}, 2(1 + \frac{a}{a+b})\} \) in this case.

Case 2. \(t1 > t2\). In this case, we have by Equation (3) that \(t1 \leq \frac{a+b}{a} (\sum_{J \in \sigma} p_J)\). For Greedy algorithm, the last job Jn is assigned to machine 2, and \(C_{greedy} \leq \max\{t1, t2 + \frac{p_n}{a+b} + b + p_n\} \leq \max\{\frac{a+b}{b} (\sum_{J \in \sigma} p_J), t2 + \frac{b}{a+b} + (1+p_n)\} \).

Moreover, \(C^* \geq \{t2, p_n, \frac{1}{2} \sum_{J \in \sigma} p_J\}\) by the case condition \(t1 > t2\) and Inequality (2).

\[ \frac{C_{greedy}}{C^*} \leq \max\{\frac{a+b + 2(a+b)}{bC^*} + \frac{2(a+b)}{b}, 1 + \frac{b}{a+b} + 1\}. \]

Hence, \( \lim_{C \to \infty} \frac{C_{greedy}}{C^*} = 2 + \frac{a}{a+b} \) if \(a \geq b\), and \(\lim_{C \to \infty} \frac{C_{greedy}}{C^*} = 2 + \frac{b}{a+b} \) if \(a < b\). Hence, \( \lim_{C \to \infty} \frac{C_{greedy}}{C^*} = \max\{2 + \frac{a}{a+b}, 2 + \frac{b}{a+b}\} \) in both cases.

It completes the proof.

V. THE MODEL OF TOTAL COMPLETION TIME

In this section, we consider the objective of total completion time, i.e., \(\sum_{j \in J} C_j\). We mainly prove the approximation performance of the SPT rule which schedules the shortest job to the machine with earliest available time and processes jobs one by one without idleness in between on both machines (the pseudocode as shown in Algorithm 2). The time complexity of this algorithm is \(O(n log(n))\).

Algorithm 2 Algorithm Based on SPT Rule

Require: n, p, Γ (a job input instance)
1: Sort all the jobs according to SPT rule, i.e. \(\sigma = (J[1], J[2], \cdots, J[n])\)
2: for j = 1 : n do
3: Calculate the available time of machine 1 and machine 2
4: if machine 1 is available at the earliest time then
5: Assign job j to machine 1
6: else
7: Assign job j to machine 2
8: end if
9: end for

Given any job instance \(\Gamma\), let \(\sigma\) be the corresponding SPT schedule without shift, and \(\sigma = \sigma_1 \cup \sigma_2\) were sub-schedules \(\sigma_1 = (J[1,1], J[1,2], \cdots, J[1,n_1])\) and \(\sigma_2 = (J[2,1], J[2,2], \cdots, J[2,n_2])\) contain jobs to be processed on
machines 1 and 2 respectively. Let \( n_i \) \((i = 1, 2)\) be the number of jobs in \( \sigma_i \), \( n_1 + n_2 = n \). Let \( \sigma'_i, \sigma'_i, \sigma''_i \) be the corresponding schedules for the scenario with shift activities. Then there are \(| \sum_{j \in \sigma_i} p_j | + 1 | \leq \sum_{j \in \sigma_i} p_j | b | + 1 \) and \( | \sum_{j \in \sigma''_i} p_j | a | \leq \sum_{j \in \sigma''_i} p_j | a | \) shift time intervals before the end of \( \sigma_i' \) respectively and \( \sigma''_i \), define \( \Delta_1 = (| \sum_{j \in \sigma_i} p_j | b | + 1 |)a | + 1 | b \) and \( \Delta_2 = | \sum_{j \in \sigma''_i} p_j | a |. \)

Notice that the jobs as well as their processing sequences in \( \sigma_i \) and \( \sigma_i' \) are the same for \( i = 1, 2 \), while their start and completion times are different. Let \( S_i, C_i \) be the start and respectively completion times of job \( j \) in schedule \( \sigma_i \) and \( \sigma_i' \). \( S_j, C_j' \) are that of the job in schedule \( \sigma_i' \).

Lemma 1: \( C_i[1,j] = (1 + \frac{a}{b})C_1[1,j] + a \), and \( C_i[2,j] = (1 + \frac{b}{a})C_2[2,j] \).

Proof: For any job \( j \in [1,j] \) on machine 1. Combining \( C_i[1,j] = \sum_{1 \leq \sigma \leq n} |j[a, b] + |a, b(a + b)| + b \) where \( \sigma \geq 0 \), we have for schedule \( \sigma_i' \) that \( C_i'[1,j] = \sum_{1 \leq \sigma \leq n} |j[a, b] + |a, b(a + b)| + b \).

It completes the proof.

By Lemma 1, for all the jobs on machine 1, \( \sum_{j \in \sigma_i'} C_j' = (1 + \frac{b}{a}) \sum_{j \in \sigma_i} C_j + n_1a \). For all the jobs on machine 2, we have \( \sum_{j \in \sigma''_i} C_j' = (1 + \frac{b}{a}) \sum_{j \in \sigma''_i} C_j \). Hence,

\[
\sum_{j \in \sigma_i'} C_j' \leq (1 + \max(\frac{a}{b}, \frac{b}{a})) \sum_{j \in \sigma_i} C_j + n_1a. \tag{5}
\]

Below we prove the following conclusion on the asymptotic performance of SPT rule.

Theorem 2: SPT is asymptotically \((1 + \max(\frac{a}{b}, \frac{b}{a}))\)-approximation for the model to minimize the total completion time.

Proof: For all the jobs in \( \sigma_i \), their total completion time is \( \sum_{1 \leq \sigma \leq n} C_i[1,j] = \sum_{1 \leq \sigma \leq n} (n_1 + 1 - j)p[1,j] \sum_{1 \leq \sigma \leq n} (n_1 + 1 - j) = \frac{n_1(n_1 + 1)}{2} \) because the above inequality is due to the fact that each job is of at least one unit of time length. As \( b \) is a positive constant, we conclude that the last addition item on the righthand side of Inequality (5) can be bounded with \( n_1a = o(\sum_{1 \leq \sigma \leq n} C_i[1,j]) \). Hence,

\[
\lim_{(\sum_{j \in \sigma_i} C_j) \to \infty} \frac{\sum_{j \in \sigma_i'} C_j'}{\sum_{j \in \sigma_i} C_j} \leq (1 + \max(\frac{a}{b}, \frac{b}{a})) + o(1). \tag{6}
\]

In an optimal schedule, denote by \( C_i' \) the completion time of job \( j \). We claim that \( C_i' \geq C_j \).

\[
\lim_{(\sum_{j \in \sigma_i} C_j) \to \infty} \frac{\sum_{j \in \sigma_i'} C_j'}{\sum_{j \in \sigma_i} C_j} \leq 1 + \max(\frac{a}{b}, \frac{b}{a}).
\]

It completes the proof.

VI. DISCUSSION

Different from previous studies, we model the problem with complementary non-available time periods. We find that the asymptotic and parametric approximation ratios for the two models are related to the length of the periodic available time intervals of the two machines, i.e., \( a \) and \( b \). What’s interesting is that whether the objective is to minimize the makespan or the total completion time, the closer the difference between the periodic available time interval of machine 1 and that of machine 2, the smaller the asymptotic and parametric approximation ratio. To the best of our knowledge, there is no similar conclusion in previous studies.

VII. CONCLUSION

In this work, we incorporate multitasking into parallel machine problem with periodic available time, and design approximation algorithms for different objectives. Future directions may include the discussion for non-periodic available time situation, the evaluation of more practical objectives, and the investigation of mixed integer programming model for this problem.

REFERENCES

[1] E. Chan, F. G. Van Zee, P. Bientinesi, E. S. Quintana-Orti, G. Quintana-Orti, and R. van de Geijn, “SuperMatrix: A multithreaded runtime scheduling system for algorithms-by-blocks,” in Proc. 15th ACM SIGPLAN SIGPLAN Symp. Prac. Parallel Program. (Ppopp), Salt Lake City, UT, USA, Feb. 2008, pp. 123–132.
[2] D. Coviello, A. Ichino, and N. Persico, “Time allocation and task juggling,” Amer. Econ. Rev., vol. 104, no. 2, pp. 609–623, Feb. 2014.
[3] F. Gaglioppa, L. A. Miller, and S. Benjaafar, “Multitask and multistage production planning and scheduling for process industries,” Oper. Res., vol. 56, no. 4, pp. 1010–1025, Aug. 2008.
[4] A. G. Hall, J. Y.-T. Leung, and C.-L. Li, “Multitasking via alternate and shared processing: Algorithms and complexity,” Discrete Appl. Math., vol. 208, pp. 41–58, Jul. 2016.
[5] R. M. Hierons and U. C. Turker, “Parallel algorithms for testing finite state machines: Generating UIO sequences,” IEEE Trans. Softw. Eng., vol. 42, no. 11, pp. 1077–1091, Nov. 2016.
[6] C.-J. Hsu, T. C. E. Cheng, and D.-L. Yang, “Unrelated parallel-machine scheduling with rate-modifying activities to minimize the total completion time,” Inf. Sci., vol. 181, no. 20, pp. 4799–4803, Oct. 2011.
[7] R. J. W. James and B. Almada-Lobo, “Single and parallel machine capacitated lotsizing and scheduling: New iterative MIP-based neighborhood search heuristics,” Comput. Oper. Res., vol. 38, no. 12, pp. 1816–1825, Dec. 2011.
[8] D. Jiang and J. Tan, “Scheduling with job rejection and nonsimultaneous machine availability time on unrelated parallel machines,” Theor. Comput. Sci., vol. 616, pp. 94–99, Feb. 2016.
[9] M. Ji, W. Zhang, L. Liao, T. C. E. Cheng, and Y. Tan, “Multitasking parallel-machine scheduling with machine-dependent slack due-window assignment,” Int. J. Prod. Res., vol. 57, no. 6, pp. 1667–1684, Mar. 2019.
[10] M. Liu, S. Wang, F. Zheng, and C. Chu, “Algorithms for the joint multitasking and common due date assignment problem,” Int. J. Prod. Res., vol. 55, no. 20, pp. 6052–6066, Oct. 2017.
[11] S. M. Mostafa and S. Kusakabe, “Towards reducing energy consumption using inter-process scheduling in preemptive multitasking OS,” in Proc. Int. Conf. Platform Technol. Service (PlatCom), Feb. 2016, pp. 1–6.
[12] T. G. Rogers, M. O’Connor, and T. M. Aamodt, “Learning your limit: Managing massively multithreaded caches through scheduling,” Commun. ACM, vol. 57, no. 12, pp. 91–98, Nov. 2014.
[13] S. Shakeri and R. Legendran, “A mathematical programming-based scheduling framework for multitasking environments,” Eur. J. Oper. Res., vol. 176, no. 1, pp. 193–209, Jan. 2007.
[14] L. Shen, D. Wang, and X.-Y. Wang, “Parallel-machine scheduling with rate-modifying activities to minimize the total completion time of job,” IEEE Trans. Softw. Eng., vol. 38, no. 11, pp. 123–132, Feb. 2008.
[15] F. Zheng et al.: Approximation Algorithms for Multitasking Scheduling Problems
[16] X.-Y. Wang, Z. Zhou, P. Ji, and J.-B. Wang, “Parallel machines scheduling with simple linear job deterioration and non-simultaneous machine available times,” Comput. Ind. Eng., vol. 74, pp. 88–91, Aug. 2014.

[17] B. Wardono and Y. Fathi, “A tabu search algorithm for the multi-stage parallel machine problem with limited buffer capacities,” Eur. J. Oper. Res., vol. 155, no. 2, pp. 380–401, Jun. 2004.

[18] J. M. Watson and D. L. Strayer, “Supertaskers: Profiles in extraordinary multitasking ability,” Psychonomic Bull. Rev., vol. 17, no. 4, pp. 479–485, Aug. 2010.

[19] X. Wu and A. Che, “Energy-efficient no-wait permutation flow shop scheduling by adaptive multi-objective variable neighborhood search,” Omega, vol. 94, pp. 102–117, 2020.

[20] X. Xiong, P. Zhou, Y. Yin, T. C. E. Cheng, and D. Li, “An exact branch-and-price algorithm for multitasking scheduling on unrelated parallel machines,” Nav. Res. Logistics, vol. 66, no. 6, pp. 502–516, Sep. 2019.

[21] F. Zhang, J. Cao, W. Tan, S. U. Khan, K. Li, and A. Y. Zomaya, “Evolutionary scheduling of dynamic multitasking workloads for big-data analytics in elastic cloud,” IEEE Trans. Emerg. Topics Comput., vol. 2, no. 3, pp. 338–351, Sep. 2014.

[22] Z. Zhu, F. Zheng, and C. Chu, “Multitasking scheduling problems with a rate-modifying activity,” Int. J. Prod. Res., vol. 55, no. 1, pp. 296–312, Jan. 2017.

[23] Z. Zhu, M. Liu, C. Chu, and J. Li, “Multitasking scheduling with multiple rate-modifying activities,” Int. Trans. Oper. Res., vol. 26, no. 5, pp. 1956–1976, Sep. 2019, doi: 10.1111/itor.12393.