Brief Announcement: Approximation of Scheduling with Calibrations on Multiple Machines

Lin Chen
Department of Computer Science, University of Houston
Houston, Texas, USA
chenlin198662@gmail.com

Minming Li
Department of Computing Science, City University of Hong Kong
Hong Kong SAR, China
minming.li@cityu.edu.hk

Guohui Lin
Department of Computing Science, University of Alberta
Edmonton, Alberta, Canada
guohui@ualberta.ca

Kai Wang
Department of Computing Science, City University of Hong Kong
Hong Kong SAR, China
Center for Advanced Studies in Management, HHL Leipzig Graduate School of Management
Leipzig, Saxony, Germany
kai.wang@my.cityu.edu.hk

ABSTRACT

We study the scheduling problem with calibrations. In 2013, Bender et al. (SPAA '13) proposed a theoretical framework for the problem. Jobs of unit processing time with release times and deadlines are to be scheduled on parallel identical machines. The machines need to be calibrated to run jobs while a single calibration remains valid on a machine only for a time period of length T. The objective is to find a schedule that completes all jobs within their timing constraints and minimizes the total number of calibrations. In this paper, we aim to design an approximation algorithm to solve the problem. We propose a dynamic programming algorithm with polynomial running time when the number of machines is constant. In addition, we give a PTAS when the number of machines is input.

CCS CONCEPTS

• **Theory of computation → Design and analysis of algorithms**; **Approximation algorithms analysis**; **Scheduling algorithms**;

KEYWORDS

Approximation Algorithm, Scheduling, Calibration, PTAS

1 INTRODUCTION

In this paper, we study the scheduling problem on multiple machines with calibrations where a machine must be calibrated before it runs a job. When the machine is calibrated at time t, it stays calibrated for a time period of length T, after which it must be recalibrated to continue running jobs. We refer to the time interval $[t, t + T]$ as the *calibration interval* and no job can be started or be processed outside the calibration intervals on a machine. We are given a set of n jobs $J = \{1, 2, ..., n\}$, where each job $j \in J$ has release time r_j, deadline d_j and processing time p_j where $p_j = 1$. We have m identical parallel machines which can be trusted to run a job only when calibrated. The objective is to find a schedule that completes all jobs within their timing constraints such that the number of calibrations is minimized.

This problem is firstly introduced and studied by Bender et al. [2], they proposed a greedy, optimal, polynomial-time algorithm called *Lazy-Binning* for the single machine case and showed that the extended Lazy-Binning algorithm on multiple machines yields 2-approximation, while the complexity status of the problem still remains open. Later on, Fineman and Sheridan [4] considered the non-preemptive jobs with non-unit processing times and generalized the problem with the consideration of resource-augmentation [5]. In 2017, Angel et al. [1] developed different results for several generalizations of this calibration minimization scheduling problem, including heterogeneous calibrations and calibration set-up time. Chau et al. [3] worked on the trade-off between weighted flow time and calibration cost for jobs of unit processing time. They integrated these two criteria in the objective function and gave several online approximation algorithms with constant approximation ratio on different settings of single or multiple machines for weighted or unweighted jobs, as well as a dynamic programming algorithm for the offline problem. Most recently, Wang [6] worked on the scheduling problem on a single machine with the consideration of both calibration and time slot cost where a certain time-dependent cost will be incurred if a time slot is occupied with some job. They considered the objective of minimizing total time slot cost with the budget of a certain number of calibrations and proposed several dynamic programming algorithms for different variants of the problem.

In this paper, we first propose a dynamic programming approach, which has polynomial running time when m is a constant. Based on the dynamic programming approach, we give a polynomial time approximation scheme (PTAS).

SPAA '19, June 22–24, 2019, Phoenix, AZ, USA

ACM ISBN 978-1-4503-6184-2/19/06.
https://doi.org/10.1145/3323165.3323173

2 PRELIMINARIES

We denote the time interval $(t-1, t]$ as *time slot* t. We sort the jobs in non-decreasing order of their deadlines, and non-decreasing order of release times if two or more jobs have the same deadline. We assume that all the input parameters are non-negative integers, in other words we only consider the feasible schedule in which the starting times of the calibrations and jobs are integers.

Definition 2.1. Let $\Psi = \bigcup_{j\in J, s\in[0,n]}\{d_j - s\}, \Phi = \bigcup_{j\in J, t\in\Psi, s\in[0,n]} \{r_j + s, t + s\}$ and $\Phi(j) = \{t \mid r_j < t \le d_j,\ t \in \Phi\},\ \forall j \in J$.

Lemma 2.2. *([1]) There exists an optimal schedule such that (i) each calibration starts at a time in* Ψ. *(ii)* $\forall j \in J$, *job* j *finishes at a time in* $\Phi(j)$.

Lemma 2.2 is summarized from the work by Angel et al. [1], which shows that there exists an optimal schedule such that the starting time of any calibration takes value from set Ψ and any job completion time takes value from set Φ. Also, by adopting the analysis from [1], one would find that $|\Psi| = O(n^2)$ and $|\Phi| = O(n^2)$.

Lemma 2.3. *([6]) There exists an optimal schedule such that for any two jobs* i, j *with* $i < j$ *(i.e.* $d_i \le d_j$*), if* $r_i \le t_j$ *then* $t_i \le t_j$ *where* t_i, t_j *are the corresponding starting times of job* i *and* j *in the optimal schedule respectively.*

Lemma 2.3 plays an important role in designing a dynamic programming algorithm and is referred from the work by Wang [6].

3 DYNAMIC PROGRAMMING APPROACH

In this section, we aim to design a dynamic program to solve the problem. In order to mark the calibrations on each machine that cover time slot t, we use a vector $\boldsymbol{\gamma} = \langle \gamma_1, \gamma_2, ..., \gamma_m\rangle$ to represent the starting times of these calibrations where $\gamma_k \in \{\text{NUL}\}\cup(\Psi\cap[t-T, t))$ indicates the starting time of the calibration on machine k and NUL represents the situation that the machine is not calibrated at time slot t. Let $\Gamma(t) = \{\langle \gamma_1, \gamma_2, ..., \gamma_m\rangle \mid \gamma_k \in \{\text{NUL}\}\cup(\Psi\cap[t-T, t)),\ \forall k \in [1, m]\}$ be the set of all possible vectors, with respect to time slot t. We use $\delta(\boldsymbol{\gamma})$ to indicate the number of valid calibrations of vector $\boldsymbol{\gamma}$, i.e. $\delta(\boldsymbol{\gamma}) = \sum_{\gamma_k\neq\text{NUL}, k\in[1,m]} 1$.

Definition 3.1. Given $j \in J,\ t_1 \in \Phi,\ t_2 \in \Phi$, let $J(j, t_1, t_2) = \{i \mid i \le j, r_i \in [t_1, t_2), i \in J\}$ be the subset of jobs whose indexes are at most j and release times are between t_1 and t_2. Let $f(j, t_1, t_2, q, \boldsymbol{u}, \boldsymbol{v})$ be the minimum number of additional calibrations needed in the partial schedule that schedules jobs $J(j, t_1, t_2)$ on m machines where $\boldsymbol{u} = \langle u_1, u_2, ..., u_m\rangle, \boldsymbol{u} - T \in \Gamma(t_1),\ \boldsymbol{v} = \langle v_1, v_2, ..., v_m\rangle \in \Gamma(t_2),\ q \in [0, m]$ on the condition that

i.) all jobs from $J(j, t_1, t_2)$ are scheduled during time interval $[t_1, t_2)$.
ii.) time intervals $[t_1, u_k)$ and $[v_k, t_2)$ have already been calibrated on machine k, $\forall k \in [1, m]$.
iii.) q other jobs (not from $J(j, t_1, t_2)$) have already been assigned to time slot t_2.

In Definition 3.1, we use vector $\boldsymbol{u}$ (resp. $\boldsymbol{v}$) to mark the ending times (resp. starting times) of the calibrations that cross the boundary of time interval $[t_1, t_2)$, i.e. that cover time slot t_1 (resp. t_2), where $\boldsymbol{u} - T \in \Gamma(t_1)$ (resp. $\boldsymbol{v} \in \Gamma(t_2)$).

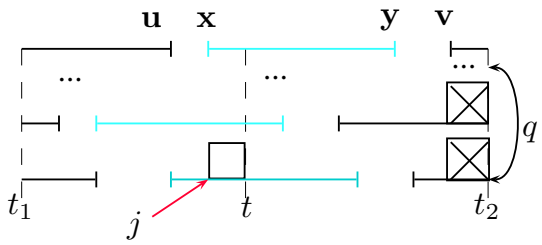

Figure 1: An illustration for the dynamic programming approach in Proposition 1.

The high level idea of the dynamic programming approach is to schedule job j at time slot t (test every possibility) and then divide the problem of scheduling jobs $J(j-1, t_1, t_2)$ over time interval $[t_1, t_2)$ into two sub-problems: scheduling jobs $J(j-1, t_1, t)$ and jobs $J(j-1, t, t_2)$ over intervals $[t_1, t)$ and $[t, t_2)$ respectively. The correctness of dividing sub-problems is proved based on Lemma 2.3.

Moreover, we open additional calibrations on the machines to cover time slot t (we do it only once for t) and we need to mark the calibration starting times on each machine that cover time slot t in the sub-problems. Specifically, we use vector $\boldsymbol{x} = \langle x_1, x_2, ..., x_m\rangle \in \Gamma(t)$ to indicate the starting times of the additional calibrations on each machine and correspondingly $\boldsymbol{y} = \langle y_1, y_2, ..., y_m\rangle = \boldsymbol{x} + T$ as the calibration ending times.

Proposition 1. *For the case* $J(j, t_1, t_2) = \emptyset$, $f(j, t_1, t_2, q, \boldsymbol{u}, \boldsymbol{v})$ *equals* 0 *if at least* q *machines are calibrated in time slot* t_2 *provided by* $\boldsymbol{u}$ *and* $\boldsymbol{v}$*, and otherwise* ∞. *If* $j \notin J(j, t_1, t_2)$ *we have* $f(j, t_1, t_2, q, \boldsymbol{u}, \boldsymbol{v}) = f(j-1, t_1, t_2, q, \boldsymbol{u}, \boldsymbol{v})$. *If* $j \in J(j, t_1, t_2), \Phi(j) \cap (t_1, t_2] = \emptyset$ *we have* $f(j, t_1, t_2, q, \boldsymbol{u}, \boldsymbol{v}) = \infty$. *Otherwise we have* $f(j, t_1, t_2, q, \boldsymbol{u}, \boldsymbol{v}) =$

$$\min_{t\in\Phi(j)\cap(t_1,t_2]} \begin{cases} \infty & , \text{if } t = t_2, q = m \\ f(j-1, t_1, t_2, q+1, \boldsymbol{u}, \boldsymbol{v}) & , \text{if } t = t_2, 0 < q < m \\ \min\limits_{cond.} f(j-1, t_1, t, 1, \boldsymbol{u}, \boldsymbol{v}') + & \\ f(j-1, t, t_2, q, \boldsymbol{u}', \boldsymbol{v}) + \delta(\boldsymbol{x}) & , \text{if } t < t_2 \text{ or } q = 0 \end{cases}$$

where cond. represents $\boldsymbol{x} \in \Gamma(t), \boldsymbol{y} = \boldsymbol{x} + T, \boldsymbol{u}' = \max\{\boldsymbol{y}, \boldsymbol{u} \ge t\}, \boldsymbol{v}' = \min\{\boldsymbol{x}, \boldsymbol{v} < t\}$.

4 APPROXIMATION ALGORITHM

In this section, we present a PTAS. Two calibrations are referred to as *distinct* if they have different starting times.

Theorem 4.1. *For any optimal schedule* σ *that satisfies Lemma 2.3, Lemma 2.2 and that no two calibrations overlap with each other on the same machine, it corresponds to another schedule* ϖ *of* $(1+\epsilon)$ *-approximation, in which for any time slot* t, *it satisfies that*

i.) *the number of distinct calibrations in* Q_t *is at most* $2\lceil 1/\epsilon\rceil + 1$.
ii.) *the number of calibrations in* Q_t *is at most* $m + \lfloor \epsilon m\rfloor + \lceil \epsilon m\rceil - 1$.
iii.) *the maximum number of calibrations that have the same starting time is at most* $m + \lfloor \epsilon m\rfloor$.
iv.) $|Q_t| \ge |Q'_t|$.

where Q_t *(resp.* Q'_t*) is the set containing all the calibrations that cover time slot* t *in schedule* ϖ *(resp. schedule* σ*).*

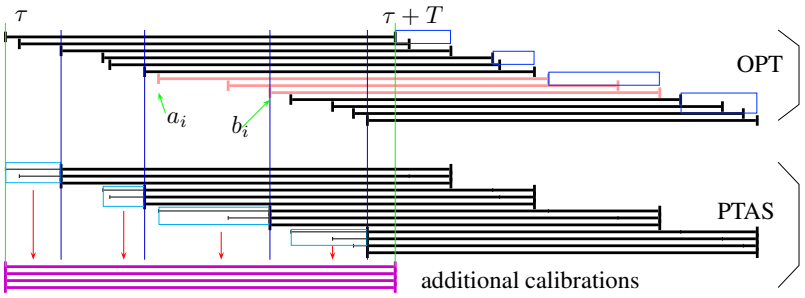

Figure 2: An illustration for Theorem 4.1 which shows the transformation of the calibrations in an optimal schedule.

We show a constructive process of schedule ϖ as follows. Consider an interval $[\tau, \tau + T)$ and suppose that there are l calibrations whose starting times belong to the interval in an optimal schedule σ. We aim to delay these l calibrations such that the number of distinct calibrations after delaying is at most $\lceil 1/\epsilon \rceil$. Moreover, we open additional calibrations with identical starting times to guarantee the last property in Theorem 4.1. The transformation process contains three steps as follows.

Step 1. (Partition) We partition the l calibrations into $\lceil 1/\epsilon \rceil$ groups such that each group contains at most $\lceil \epsilon l \rceil$ calibrations. This is feasible as $\lceil 1/\epsilon \rceil \cdot \lceil \epsilon l \rceil \geq l$.

Step 2. (Delay) For each group of calibrations, we delay all calibrations in this group so that they have the same starting time with the latest calibration in the group.

Step 3. (Augment) We start $\lfloor \epsilon l \rfloor$ additional calibrations at time t_0 where t_0 is the earliest starting time of those l calibrations.

By applying the above steps independently for every sub-interval ($[\tau, \tau + T)$) of length T, the new schedule will fulfill the properties in Theorem 4.1.

Find an approximation solution.

Based on the properties of Theorem 4.1, we present an efficient way of marking the calibrations. Let $h = 2\lceil 1/\epsilon \rceil + 1$, $m' = m + \lfloor \epsilon m \rfloor$, $\hat{m} = m + \lfloor \epsilon m \rfloor + \lceil \epsilon m \rceil - 1$, and we assume $m' < n$ as we only open less than n calibrations in the approximation schedule.

Definition 4.2. A *configuration* is defined as a pair of vectors $\langle \boldsymbol{\alpha}, \boldsymbol{\eta} \rangle$ where $\boldsymbol{\alpha} = \langle \alpha_1, \alpha_2, ..., \alpha_h \rangle$, $\boldsymbol{\eta} = \langle \eta_1, \eta_2, ..., \eta_h \rangle$ and for each $i \in [1, h]$, $\alpha_i \in \{\text{NUL}\} \cup \Psi$ indicates the starting time of a calibration, $\eta_i \in [1, m']$ indicates the number of the calibrations that share the same starting time α_i. We define $\mathcal{A} = \{\langle \alpha_1, \alpha_2, ..., \alpha_h \rangle \mid \alpha_i \in \{\text{NUL}\} \cup \Psi, \forall i \in [1, h]|\}$ to be the set of all possible vectors $\boldsymbol{\alpha}$. Given time slot t, we define $\mathcal{A}(t) = \{\langle \alpha_1, \alpha_2, ..., \alpha_h \rangle \mid \alpha_i \in \{\text{NUL}\} \cup (\Psi \cap [t - T, t)), \forall i \in [1, h]\}$, where α_i indicates the starting time of a calibration which covers time slot t. Let $\mathcal{B} = \{\boldsymbol{\eta} \mid \sum_{i=0}^{h} \eta_i \leq \hat{m}, \eta_i \in [1, m'], \forall i \in [1, h]\}$ be the set of all possible vectors $\boldsymbol{\eta}$.

Previously, for the calibrations that cover time slot t, we use vector $\boldsymbol{\gamma} = \langle \gamma_1, \gamma_2, ..., \gamma_m \rangle \in \Gamma(t)$ to mark the calibration starting time on each machine. In the modified dynamic programming approach, we use configurations to mark the distinct calibration starting times and the number of occurrence of each starting time. Although calibrations might overlap with each other, we guarantee that for any time slot t, the number of jobs that are scheduled in time slot t is no more than m. Note that in total we have at most $(m')^h |\Psi|^h$ configurations as $|\mathcal{A}| \leq |\Psi|^h$ and $|\mathcal{B}| \leq (m')^h \leq n^h$, which implies that the total number of possible configurations is polynomial in n. In summary, we replace vector $\boldsymbol{\gamma} = \langle \gamma_1, \gamma_2, ..., \gamma_m \rangle$ of set $\Gamma(t)$ by configuration $\langle \boldsymbol{\alpha}, \boldsymbol{\eta} \rangle$, and apply the dynamic program in Proposition 1 to obtain an approximated solution.

5 CONCLUSION

We studied the scheduling problem with calibrations on multiple machines where we consider the schedule of unit processing time jobs with release times and deadlines such that the total number of calibrations is minimized. We proposed a dynamic programming approach to solving the problem with running time $O(mn^{7+6m})$. Moreover, we presented a PTAS, which has running time $O(mn^{16+18\lceil 1/\epsilon \rceil})$. It is challenging to tackle the open problem proposed by Bender et al. [2] about the complexity status on multiple machines with jobs of unit processing times.

ACKNOWLEDGMENTS

The work described in this paper was supported by a grant from Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CityU 11268616) and Project 11771365 supported by NSFC and a grand from NSF 1756014.

REFERENCES

[1] Eric Angel, Evripidis Bampis, Vincent Chau, and Vassilis Zissimopoulos. 2017. On the Complexity of Minimizing the Total Calibration Cost. In *International Workshop on Frontiers in Algorithmics*. Springer, 1–12. https://doi.org/10.1007/978-3-319-59605-1_1

[2] Michael A. Bender, David P. Bunde, Vitus J. Leung, Samuel McCauley, and Cynthia A. Phillips. 2013. Efficient Scheduling to Minimize Calibrations. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '13)*. ACM, New York, NY, USA, 280–287. https://doi.org/10.1145/2486159.2486193

[3] Vincent Chau, Minming Li, Samuel McCauley, and Kai Wang. 2017. Minimizing Total Weighted Flow Time with Calibrations. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '17)*. ACM, New York, NY, USA, 67–76. https://doi.org/10.1145/3087556.3087573

[4] Jeremy T. Fineman and Brendan Sheridan. 2015. Scheduling Non-Unit Jobs to Minimize Calibrations. In *Proceedings of the 27th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '15)*. ACM, New York, NY, USA, 161–170. https://doi.org/10.1145/2755573.2755605

[5] Bala Kalyanasundaram and Kirk Pruhs. 2000. Speed is As Powerful As Clairvoyance. *J. ACM* 47, 4 (July 2000), 617–643. https://doi.org/10.1145/347476.347479

[6] Kai Wang. 2018. Calibration Scheduling with Time Slot Cost. In *Algorithmic Aspects in Information and Management - 12th International Conference, AAIM 2018, Dallas, TX, USA, December 3-4, 2018, Proceedings (Lecture Notes in Computer Science)*, Shaojie Tang, Ding-Zhu Du, David L. Woodruff, and Sergiy Butenko (Eds.), Vol. 11343. Springer, 136–148. https://doi.org/10.1007/978-3-030-04618-7_12