




# Revisit the Scheduling Problem with Calibrations

Lin Chen  

Department of Computer Science, Zhejiang University, China

Yixiong Gao   

Department of Computer Science, City University of Hong Kong, Kowloon, China

Minming Li   

Department of Computer Science, City University of Hong Kong, Kowloon, China

Guohui Lin   

Department of Computing Science, University of Alberta, Edmonton, Canada

Kai Wang  

Department of Computer Science, City University of Hong Kong, Kowloon, China

---

## Abstract

The research about scheduling with calibrations was initiated from the Integrated Stockpile Evaluation (ISE) program which tests nuclear weapons periodically. The tests for these weapons require calibrations that are expensive in the monetary sense. This model has many industrial applications where the machines need to be calibrated periodically to ensure high-quality products, including robotics and digital cameras. In 2013, Bender et al. (SPAA '13) proposed a theoretical framework for the ISE problem. In this model, a machine can only be trusted to run a job when it is calibrated and the calibration remains valid for a time period of length  $T$ , after which it must be recalibrated before running more jobs. The objective is to find a schedule that completes all jobs by their deadlines and minimizes the total number of calibrations. In this paper, we study the scheduling problem with calibrations on multiple parallel machines where we consider unit-time processing jobs with release times and deadlines. We propose a dynamic programming algorithm with polynomial running time when the number of machines is constant. Then, we propose another dynamic programming approach with polynomial running time when the length of the calibrated period is constant. Also, we propose a PTAS, that is, for any constant  $\epsilon > 0$ , we give a  $(1 + \epsilon)$ -approximation solution with  $m$  machines.

**2012 ACM Subject Classification** Theory of computation → Scheduling algorithms

**Keywords and phrases** Approximation Algorithm, Scheduling, Calibration, Resource Augmentation

**Digital Object Identifier** 10.4230/LIPIcs.ISAAC.2024.20

## 1 Introduction

The original motivation for scheduling with calibrations came directly from the Integrated Stockpile Evaluation (ISE) program which tests nuclear weapons periodically [6]. The tests for these weapons require calibrations that are expensive. This motivation can be extended to the scenarios where the machines need to be calibrated periodically to ensure high-quality products, such as robotics and digital cameras [17, 4, 27]. In this model, 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 run more jobs. We refer to the time interval  $[t, t + T]$  as the *calibration interval* and no job can be started or be processed on a machine outside the times sitting in a calibration interval. In the ideal model, calibrating a machine is instantaneous, meaning that the machine can run a job immediately after being calibrated and the machine can switch from uncalibrated to calibrated status instantaneously. The objective is to assign the jobs to the machines using the minimum number of calibrations.



© Lin Chen, Yixiong Gao, Minming Li, Guohui Lin, and Kai Wang;  
licensed under Creative Commons License CC-BY 4.0

35th International Symposium on Algorithms and Computation (ISAAC 2024).

Editors: Julián Mestre and Anthony Wirth; Article No. 20; pp. 20:1–20:15

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Methodologies have been introduced about performing a calibration in different scenarios [20, 23, 26], as well as determining the time period of a calibration [16, 18]. On the other hand, there is quite a lot of research work about scheduling algorithms [5]. In 2013, Bender et al. [3] proposed a theoretical framework for scheduling with calibrations. They considered unit-time jobs with release times and deadlines, aiming at minimizing the number of calibrations. In the single-machine setting, they proposed a greedy, optimal, polynomial-time algorithm called *Lazy-Binning* where the algorithm delays the start of a calibration interval for as long as possible, until delaying it further would make it impossible to find a feasible schedule. For the multiple machine setting, they proposed the *Lazy-Binning* algorithm on multiple machines and showed it is a 2-approximation algorithm, while the complexity status still remains open.

Later, Fineman and Sheridan [12] considered the non-preemptive jobs with non-unit processing times and generalized the problem with resource-augmentation [15] in the sense that an approximate solution can be obtained if we speed up the machines and/or have more machines. They worked on multiple machine setting without preemption, showed the relationship of the problem with the classical machine-minimization problem [19] and proved that if there is an  $s$ -speed  $\alpha$ -approximation algorithm for the machine-minimization problem, then it would give an  $s$ -speed  $O(\alpha)$ -machine  $O(\alpha)$ -approximation solution for the calibration minimization problem.

Angel et al. [2] developed different results on several generalizations of this problem. They considered the jobs of non-unit processing times on a single machine with preemption, and proposed an optimal greedy algorithm, which extends the idea of the *Lazy Binning* algorithm. Also, they extended the model to allow many types of calibrations where different types of calibrations have different interval lengths and costs, and proved that the problem is NP-hard. At last, they considered a more realistic case where calibrating a machine takes a fixed amount of time, and proposed a dynamic programming approach to solve the problem. Chau et al. [7] showed a 3-approximation polynomial time algorithm when jobs have unit processing times. Also, they showed a  $(3/(1 - \varepsilon))$ -approximation pseudo-polynomial time algorithm and a  $(18/(1 - \varepsilon))$ -approximation polynomial time algorithm for the arbitrary processing time case. Chau et al. [9] showed that the scheduling problem with batch calibrations can be solved in polynomial time. Also, they proposed some fast approximation algorithms for several special cases. Chen and Zhang [11] considered online scheduling with calibration while calibrating a machine will require certain time units. They gave an asymptotically optimal algorithm for this problem when all the jobs have unit processing times, and improved the competitive ratio for the special case that calibrating a machine is instantaneous.

While the above works are about minimizing the number of calibrations, Chau et al. [8] worked on the trade-off between weighted flow time and calibration cost for unit-time jobs. They integrated the objective function with these two criteria, and gave several online approximation results on different settings and also a dynamic programming method for the offline problem. Wang [24] worked on the scheduling of minimizing total time slot cost with calibration requirements. They considered jobs of identical processing times, and proposed three different dynamic programs for different scenarios. Chen et al. [10] studied the scheduling problem with multiple types of calibrations and proposed constant approximation algorithms for several types of calibrations.

In this paper, we work on the original problem proposed by Bender et al. [3] in which jobs have unit processing times with release times and deadlines. We abbreviate this problem as  $P|r_j, p_j = 1, d_j, T|\#\text{Calibrations with machine calibrations}$ , where we adopt the classical three-field notation in scheduling of Graham *et al.* [13]. We propose two dynamic

programming approaches to solve the problem with polynomial running time when (i) the number of machines is constant, (ii) or the valid length of a calibration is constant. Moreover, when the number of machine and the valid length of a calibration are both inputs, we present a PTAS. Our results are summarized in Table 1. It is worth noting that the currently best result for this problem is a 2-approximation algorithm by Bender et al. [3], and the complexity status still remains open.

■ **Table 1** A summary of the results of scheduling with calibrations on multiple machines.

Algorithm	Approximation Ratio	Time Complexity	Remark
Section 3	1	$O(n^{8+6m})$	$m$ is constant, $T$ is input
Section 4	1	$O(n^8 m^{3T})$	$m$ is input, $T$ is constant
Section 5	$1 + \epsilon$	$O(n^{17+18\lceil 1/\epsilon \rceil})$	Both $m, T$ are input
Bender et al. [3]	2	$\text{Poly}(n, m)$	Lazy-binning approach

Transforming a dynamic programming formulation into a PTAS (polynomial time approximation scheme) has been studied decades ago [14, 21]. Woeginger [25] proposed a general technique and gave some conditions to identify whether a dynamic programming formulation could be transformed into an FPTAS. Schuurman and Woeginger [22] summarized the methods into three main categories, structuring the input, structuring the output and structuring the execution of an algorithm.

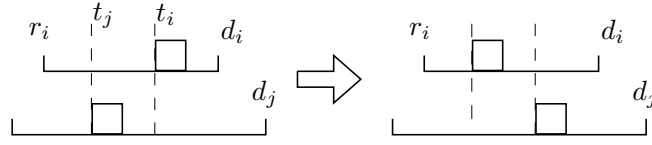
For our problem, it is not straightforward to directly apply Schuurman and Woeginger’s [22] method to get an FPTAS. Therefore, in this paper, we show a different approach to transform the dynamic program formulation into a PTAS, which lies in the category of structuring the output. In Section 2 we present the structure of an optimal schedule of this problem. Then in Section 3 we propose a dynamic programming approach to solve the problem. In Section 4 we propose another dynamic programming approach for the case when  $T$  is constant. In Section 5 we give a PTAS algorithm. We conclude our results in Section 6.

## 2 Formulation

We are given a set  $J$  of  $n$  jobs, where each job  $j \in J$  has release time  $r_j$ , deadline  $d_j$  and processing time  $p_j = 1$ . We have  $m$  identical parallel machines which can be trusted to run a job only when calibrated. The calibration remains valid for a time period of length  $T$ . The objective is to find a schedule that completes all jobs before their deadlines such that the number of calibrations is minimized. We assume that all the input are non-negative integers.

A feasible solution includes the schedule of calibrations (i.e., when to start a calibration on each machine) and the schedule of jobs (i.e., when and on which machine to start a job). We assume all the inputs are integers and both calibrations and jobs should start at times which are also integers. We denote the time interval  $(t - 1, t]$  as *time slot*  $t$  and  $t$  is called *active* if some job is scheduled in this time slot. We sort the jobs by non-decreasing order of their deadlines, and non-decreasing order of release times if two or more jobs have the same deadline. The jobs are indexed from 1 to  $n$ , and any job  $j \in J$  has index  $j \in [1, n]$ . As a matter of fact, once the schedule of calibrations is fixed, the schedule of jobs can be obtained by applying the classical *Earliest-Deadline-First* (EDF) scheduling algorithm, in which for any time slot, the job of the earliest deadline has the highest priority to be considered to schedule whenever a machine is available (i.e., calibrated).

## 20:4 Revisit the Scheduling Problem with Calibrations



■ **Figure 1** An illustration for Lemma 1.

► **Lemma 1 (EDF).** *There exists an optimal schedule such that for any two jobs  $i, j$  with  $i < j$ , if  $r_i \leq t_j$  then  $t_i \leq t_j$  where  $t_i, t_j$  are the corresponding starting times of job  $i$  and  $j$  in the optimal schedule respectively.*

**Proof.** As job  $i$  has smaller index than job  $j$ , i.e.,  $i < j$ , we have  $d_i \leq d_j$ . Suppose  $t_i > t_j$  in the optimal schedule, and then we have  $r_i \leq t_j < t_i < d_i \leq d_j$ , which implies that by swapping the schedule of the two jobs  $i$  and  $j$  (schedule job  $i$  at time  $t_j$  and job  $j$  at time  $t_i$ ), the new schedule is still feasible and follows EDF scheduling policy. In order to prove that another optimal schedule can be obtained after a finite number of the above swapping process, we only need to show that a certain value of the schedule decreases after the swapping process. Especially, after swapping, the value  $(i - t_i)^2 + (j - t_j)^2$  strictly decreases since  $(i - t_i)^2 + (j - t_j)^2 > (i - t_j)^2 + (j - t_i)^2$ . Therefore, we will eventually obtain an optimal schedule satisfying the statement. ◀

► **Definition 2.** 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  $\Phi(j) = \{t \mid r_j < t \leq d_j, t \in \Phi\}$ ,  $\forall j \in J$ .

One would find that  $|\Psi| = O(n^2)$ . Also  $|\Phi| = O(n^2)$  since for each  $t \in \Psi$  we have  $t = d_j - s$  for some  $j \in J, s \in [0, n]$  and the number of possible values of  $d_j - s + s'$  for  $s' \in [0, n]$  is bounded by  $O(n^2)$ . The following lemma is from the work by Angel et al. [1].

► **Lemma 3 ([1]).** *There always exists an optimal schedule such that*

- i.) *each calibration starts at a time in  $\Psi$ .*
- ii.)  *$\forall j \in J$ , job  $j$  finishes at a time in  $\Phi(j)$ .*

### 3 Dynamic Programming Approach

In this section we introduce a dynamic programming approach to solve the problem. Assume that the calibrations on the same machine never overlap with each other and we look for the optimal solution which follows EDF scheduling policy. We focus on the problem within a specific time interval  $[t_1, t_2)$  and consider the jobs that are released during this interval, where  $t_1, t_2$  are the possible job completion times. Lemma 1 shows a very important property that once job  $j$  is scheduled at time slot  $t$  in the optimal solution, the remaining jobs (whose index is less than  $j$ ) could be partitioned into two groups such that they must be scheduled during time intervals  $[t_1, t)$  and  $[t, t_2)$  in the optimal solution, respectively (more details in later analysis). Therefore, we could split the problem into sub-problems. In the sub-problem, we need to mark the calibrations that cross the boundary of the time interval  $[t_1, t_2)$ , where we use vectors defined in the following.

Given time slot  $t$ , some machines may be calibrated at slot  $t$  and others may not. We use notation NUL to represent the situation that the machine is not calibrated at some time slot (NUL represents NULL in programming). In order to mark the calibrations on each machine that cover time slot  $t$  (there could be at most  $m$  such calibrations), we use a vector  $\gamma = \langle \gamma_1, \gamma_2, \dots, \gamma_m \rangle$  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  $k$ . Let  $\Gamma(t) = \{\langle \gamma_1, \gamma_2, \dots, \gamma_m \rangle \mid \gamma_k \in \{\text{NUL}\} \cup (\Psi \cap [t - T, t]), \forall k \in [1, m]\}$  be the set of all possible vectors, with respect to time slot  $t$ .

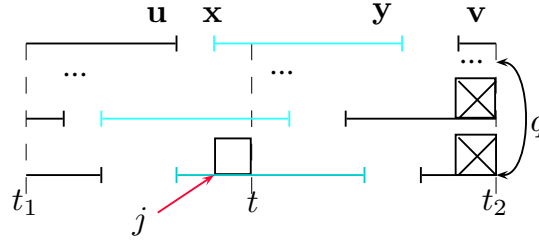
► **Definition 4.** Let  $J(j, t_1, t_2)$ ,  $\forall j \in J$  be the subset of jobs whose index is at most  $j$  and release time is between  $t_1$  and  $t_2$ , i.e.,  $J(j, t_1, t_2) = \{i \mid i \leq j, r_i \in [t_1, t_2], i \in J\}$

► **Definition 5.** Let  $f(j, t_1, t_2, q, \mathbf{u}, \mathbf{v})$  be the minimum number of calibrations to schedule jobs  $J(j, t_1, t_2)$  on  $m$  machines where  $\mathbf{u} = \langle u_1, u_2, \dots, u_m \rangle$ ,  $\mathbf{u} - T \in \Gamma(t_1)$ ,  $\mathbf{v} = \langle v_1, v_2, \dots, v_m \rangle \in \Gamma(t_2)$ ,  $t_1 \in \Phi$ ,  $t_2 \in \Phi$ ,  $q \in [0, m]$  on the condition that

- i.) jobs  $J(j, t_1, t_2)$  are only 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 the definition, we use vector  $\mathbf{u}$  (resp.  $\mathbf{v}$ ) to mark the calibration ending times (resp. starting times) of the calibrations that cross the boundary of interval  $[t_1, t_2]$ , i.e., covering time slot  $t_1$  (resp.  $t_2$ ), where  $\mathbf{u} - T \in \Gamma(t_1)$  (resp.  $\mathbf{v} \in \Gamma(t_2)$ ). We use parameter  $q$  to reserve  $q$  machines at slot  $t_2$  in order to schedule the jobs (not from  $J(j, t_1, t_2)$ ) that are assigned to slot  $t_2$ .

We consider the starting time of job  $j$  and suppose job  $j$  is scheduled at time slot  $t$  in the optimal schedule (refer to Figure 2). If a job from  $J(j - 1, t_1, t_2)$  is released before  $t$ , then it must be scheduled before (or at) time slot  $t$  by Lemma 1. Therefore, the remaining jobs  $J(j - 1, t_1, t_2)$  can be partitioned into two groups: jobs  $J(j - 1, t_1, t)$  and jobs  $J(j - 1, t, t_2)$ . For jobs  $J(j - 1, t_1, t)$ , they will not be scheduled after time  $t$  as argued, and for jobs  $J(j - 1, t, t_2)$  they cannot be scheduled before  $t$  because of the job release time. Hence the original problem could be divided into two sub-problems. Moreover, we have to enumerate the calibrations (i.e., the calibration starting times) on each machine that cover time slot  $t$  in the optimal schedule, in order to schedule job  $j$  at time slot  $t$ . Specifically, we use vector  $\mathbf{x} = \langle x_1, x_2, \dots, x_m \rangle \in \Gamma(t)$  to indicate the starting times of the calibrations on all machines and correspondingly  $\mathbf{y} = \langle y_1, y_2, \dots, y_m \rangle = \mathbf{x} + T$  as the calibration ending times.



■ **Figure 2** An illustration for Proposition 7.

In the following, we define the operations that is related to NUL.

► **Definition 6.** For any  $x \in \mathbb{R}$ , we define the min and max functions  $\min\{\text{NUL}, x\}$ ,  $\min\{x, \text{NUL}\}$ ,  $\max\{\text{NUL}, x\}$ ,  $\max\{x, \text{NUL}\}$  to be  $x$ , the operations  $x + \text{NUL}$ ,  $x - \text{NUL}$ ,  $\text{NUL} - \text{NUL}$ ,  $\min\{\text{NUL}, \text{NUL}\}$  to be  $\text{NUL}$ , and the intervals  $[x, \text{NUL}]$ ,  $[\text{NUL}, x]$  to be  $\emptyset$ .

We define min (or max, analogously) function on two vectors  $\boldsymbol{\gamma}, \boldsymbol{\lambda}$  to be  $\boldsymbol{\beta} = \min\{\boldsymbol{\gamma}, \boldsymbol{\lambda}\}$  where  $\beta_k = \min\{\gamma_k, \lambda_k\}$ ,  $\forall k \in [1, m]$  (recall that  $\gamma_k$  or  $\lambda_k$  might be NUL). We define operator  $+$  (or  $-$ , analogously) on a vector  $\boldsymbol{\lambda}$  and a number  $x$  to be  $\boldsymbol{\beta} = \boldsymbol{\lambda} + x$  where  $\beta_k = \lambda_k + x$ . And we define operator  $<$  (or  $\leq, >, \geq$ , analogously) to be  $\boldsymbol{\beta} = (\boldsymbol{\lambda} < x)$  where  $\beta_k = \lambda_k$  if  $\lambda_k \neq \text{NUL}$ ,  $\lambda_k < x$  and otherwise  $\beta_k = \text{NUL}$ .

## 20:6 Revisit the Scheduling Problem with Calibrations

Let function  $\delta(\boldsymbol{\lambda}) = \sum_{\lambda_k \neq \text{NUL}, k \in [1, m]} 1$  on vector  $\boldsymbol{\lambda}$  be the function that indicates the number of real values in vector  $\boldsymbol{\lambda}$ .

► **Proposition 7.** For the case  $J(j, t_1, t_2) = \emptyset$ , we set  $f(j, t_1, t_2, q, \mathbf{u}, \mathbf{v})$  to be 0 if at least  $q$  machines are calibrated at time slot  $t_2$  providing  $\mathbf{u}$  and  $\mathbf{v}$ , and otherwise  $\infty$ . If  $j \notin J(j, t_1, t_2)$  we have  $f(j, t_1, t_2, q, \mathbf{u}, \mathbf{v}) = f(j - 1, t_1, t_2, q, \mathbf{u}, \mathbf{v})$ . If  $j \in J(j, t_1, t_2)$  and  $\Phi(j) \cap (t_1, t_2] = \emptyset$ , we have  $f(j, t_1, t_2, q, \mathbf{u}, \mathbf{v}) = \infty$ . Otherwise we have  $f(j, t_1, t_2, q, \mathbf{u}, \mathbf{v}) =$

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

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

**Proof.** For the base cases, if  $J(j, t_1, t_2) = \emptyset$ , no job needs to be scheduled, hence we only need to guarantee that at least  $q$  machines are calibrated at time slot  $t_2$ . If  $j \notin J(j, t_1, t_2)$ , we would have  $J(j, t_1, t_2) = J(j - 1, t_1, t_2)$ . If  $j \in J(j, t_1, t_2)$ ,  $\Phi(j) \cap (t_1, t_2] = \emptyset$ , it is impossible to schedule job  $j$  during interval  $(t_1, t_2]$ .

In the dynamic programming equation, we allow calibrations to overlap with each other on the same machine, and we guarantee that for each time slot, the number of jobs that are assigned to this time slot is at most the number of machines that are calibrated during this time slot. Firstly, we try every possibility (specifically, job starting time) to schedule job  $j$ . Secondly, we follow the scheduling policy that whenever a time slot is active, we try every possibility of the calibrations (specifically, calibration starting times) on each machine that cover this time slot. Depending on the time slot  $t$  where job  $j$  is scheduled, we divide the analysis into three cases.

**Case 1)**  $t = t_2, q = m$ . In this case, there are already  $q$  jobs assigned to time slot  $t_2$ . Therefore it is infeasible to assign job  $j$  to time slot  $t_2$ .

**Case 2)**  $t = t_2, 0 < q < m$ . In this case, job  $j$  is assigned to time slot  $t_2$ . Since  $q > 0$ , i.e., some job is already scheduled at time slot  $t_2$ , by our approach the calibration decision on time slot  $t_2$  is already made, which means that we do not need to enumerate again the calibrations that cover time slot  $t_2$ . Therefore we just schedule job  $j$  at slot  $t_2$  and recurse to the sub-problem  $f(j - 1, t_1, t_2, q + 1, \mathbf{u}, \mathbf{v})$  where we reserve  $q + 1$  machines at time slot  $t_2$ .

**Case 3)**  $t < t_2$  or  $q = 0$ . If  $t < t_2$ , time slot  $t$  is not yet active as we only reserve time slot  $t_2$  for other jobs. If  $t = t_2, q = 0$ , no job is reserved at time slot  $t_2$  by definition. Therefore, in this case time slot  $t$  is not yet active and we enumerate the calibrations that cover time slot  $t$ , i.e., vector  $\mathbf{x} \in \Gamma(t)$ . Once we fix time slot  $t$  and vector  $\mathbf{x}$ , we partition the remaining jobs  $J(j - 1, t_1, t_2)$  into two groups: jobs  $J(j - 1, t_1, t)$  and jobs  $J(j - 1, t, t_2)$ . Because jobs in  $J(j - 1, t_1, t)$  will not be scheduled after time  $t$  in the optimal solution by Lemma 1 and jobs in  $J(j - 1, t, t_2)$  cannot be scheduled before  $t$  because of the job release time, the two groups of jobs must be scheduled during interval  $[t_1, t)$  and  $[t, t_2)$  respectively. The only issue left is to determine the calibrations that cross the boundary of the intervals. For the calibrations that intersect with interval  $[t_1, t)$  and cover time slot  $t$ , they must come from the calibrations with starting time  $\mathbf{x}$  or  $\mathbf{v}$ . Hence we define  $\mathbf{v}' = \min\{\mathbf{x}, \mathbf{v} < t\}$  to include as many calibrated slots on each machine as possible for interval  $[t_1, t)$  and define the first sub-problem to be  $f(j - 1, t_1, t, 1, \mathbf{u}, \mathbf{v}')$ , in which we reserve one machine at time slot  $t$  for the schedule of job  $j$ . Note that  $\mathbf{v}'$  follows the definition of the sub-problem, i.e.,  $\mathbf{v}' \in \Gamma(t)$ . Symmetrically, we

define vector  $\mathbf{u}' = \max\{\mathbf{y}, \mathbf{u} \geq t\}$  to include as many calibrated slots on each machine as possible for interval  $[t, t_2)$  and we have  $\mathbf{u}' - T \in \Gamma(t)$ . We define the sub-problem to be  $f(j-1, t, t_2, q, \mathbf{u}', \mathbf{v})$ . As we have tried every possibility of time slot  $t$ , vector  $\mathbf{x}$ , at least one try is the same as the optimal solution. Hence, we obtain two partial schedules from two sub-problems respectively and then schedule job  $j$  at time slot  $t$ . Job  $j$  is feasible because the first sub-problem has reserved a time slot for job  $j$  at slot  $t$  by definition.  $\blacktriangleleft$

**Time Complexity.** The dynamic program has table size  $O(n^2|\Phi|^2|\Psi|^{2m})$ . Constructing the solution from the sub-problems takes  $O(|\Phi||\Psi|^m)$  steps. In total, the running time is  $O(n^2|\Phi|^3|\Psi|^{3m}) = O(n^{8+6m})$ . When  $m$  is constant, i.e., the number of machines is constant, our dynamic programming approach has polynomial running time.

#### 4 When $T$ is Constant

The algorithm in Section 3 is exponential on the number of machines (i.e.,  $m$ ). When  $m$  is input and  $T$  is constant, we introduce another dynamic programming approach in this section with polynomial running time. Since all jobs have unit processing times, the scheduling between two different time slots is independent once the calibration scheme and the starting time of each job are determined. That is, for a certain time slot  $t$ , a job processed on this slot and any two available (i.e., calibrated) machines on this slot, it makes no difference to schedule the job on either of the two machines for the schedule of any other time slot  $t'$ . This inspires us that we do not need to distinguish machines, but only need to distinguish calibrations with different starting times. When  $T$  is constant, we can use this to optimize the table's cardinality in the dynamic programming approach to a polynomial of  $n$  and  $m$ .

**Definition 8.** Assume that the calibrations on the same machine never overlap. To mark the status of calibrations at time slot  $t$ , we use a vector  $\mathbf{c}_t = \langle c_{t,1}, c_{t,2}, \dots, c_{t,T} \rangle$  to represent the numbers of calibrations distinguished by the number of remaining time slots that kept the machine calibrated, where  $c_{t,k} \in \{0, 1, 2, \dots, m\}$  indicates the number of calibrations that makes the machine available at time slot  $t, t+1, \dots, t+k-1$ .

We define function  $\iota$  on vector  $\mathbf{c}_t$  and an integer  $x$  as  $\iota(\mathbf{c}_t, x) = \mathbf{c}_{t+x}$  where  $c_{t+x,k} = c_{t,k-x}$  if  $k-x \in [1, T]$ , and otherwise 0,  $\forall k \in [1, T]$ . We define max function on two vectors  $\boldsymbol{\gamma}, \boldsymbol{\lambda}$  to be  $\boldsymbol{\beta} = \max\{\boldsymbol{\gamma}, \boldsymbol{\lambda}\}$  where  $\beta_k = \max\{\gamma_k, \lambda_k\}$ ,  $\forall k \in [1, T]$ .

Let  $\mathcal{C}(t) = \{\langle c_{t,1}, c_{t,2}, \dots, c_{t,T} \rangle \mid c_{t,k} \in \{0, 1, 2, \dots, m\}, \forall k \in [1, T] \wedge \sum_{k=1}^T c_{t,k} \leq m\}$  be the set of all possible vectors, with respect to time slot  $t$ . One would find  $|\mathcal{C}(t)| = O(m^T)$ .

**Definition 9.** We define  $f(j, t_1, t_2, q, \mathbf{c}_{t_1}, \mathbf{c}_{t_2})$  to be the minimum number of calibrations to schedule jobs  $J(j, t_1, t_2)$  on  $m$  machines where  $\mathbf{c}_{t_1} \in \mathcal{C}(t_1), \mathbf{c}_{t_2} \in \mathcal{C}(t_2)$ ,  $t_1 \in \Phi$ ,  $t_2 \in \Phi$ ,  $q \in [0, m]$  on the condition that

- i.) jobs  $J(j, t_1, t_2)$  are only scheduled during time interval  $[t_1, t_2)$ .
- ii.) machines have already been calibrated according to  $\mathbf{c}_{t_1}$  and  $\mathbf{c}_{t_2}$ .
- iii.)  $q$  other jobs (not from  $J(j, t_1, t_2)$ ) have already been assigned to time slot  $t_2$ .

**Proposition 10.** For the case  $J(j, t_1, t_2) = \emptyset$ , we set  $f(j, t_1, t_2, q, \mathbf{c}_{t_1}, \mathbf{c}_{t_2})$  to be 0 if there are at least  $q$  available machines on time slot  $t$  (i.e.,  $q \leq \sum_{k=1}^T c_{t_2,k}$ ), and otherwise  $\infty$ . If  $j \notin J(j, t_1, t_2)$ , we have  $f(j, t_1, t_2, q, \mathbf{c}_{t_1}, \mathbf{c}_{t_2}) = f(j-1, t_1, t_2, q, \mathbf{c}_{t_1}, \mathbf{c}_{t_2})$ . If  $j \in J(j, t_1, t_2)$  and  $\Phi(j) \cap (t_1, t_2] = \emptyset$ , we have  $f(j, t_1, t_2, q, \mathbf{c}_{t_1}, \mathbf{c}_{t_2}) = \infty$ . Otherwise we have



$$f(j, t_1, t_2, q, \mathbf{c}_{t_1}, \mathbf{c}_{t_2}) = \min_{t \in \Phi(j) \cap (t_1, t_2]} \begin{cases} \infty & , \text{ if } t = t_2, q = m \\ f(j-1, t_1, t_2, q+1, \mathbf{c}_{t_1}, \mathbf{c}_{t_2}) & , \text{ if } t = t_2, 0 < q < m \\ \min_{\text{cond.}} \sum_{k=1}^T c_{t,k} & \\ + f(j-1, t_1, t, 1, \mathbf{c}_{t_1}, \dot{\mathbf{c}}_t) & \\ + f(j-1, t, t_2, q, \ddot{\mathbf{c}}_t, \mathbf{c}_{t_2}) & , \text{ if } t < t_2 \text{ or } q = 0 \end{cases}$$

where *cond.* stands for  $\dot{\mathbf{c}}_t = \max\{\iota(\mathbf{c}_{t_1}, t - t_1), \mathbf{c}_t\}$ ,  $\ddot{\mathbf{c}}_t = \max\{\iota(\mathbf{c}_{t_2}, t - t_2), \mathbf{c}_t\}$  where  $\mathbf{c}_t, \dot{\mathbf{c}}_t, \ddot{\mathbf{c}}_t \in \mathcal{C}(t)$ .

**Proof.** The structure of the proof is similar to Proposition 7. Beyond the base cases, we first try every possible starting time  $t$  to schedule job  $j$ . Then we follow the scheduling policy that whenever a time slot is active, we try every possibility of  $\mathbf{c}_t$ . Depending on the time slot  $t$  where job  $j$  is scheduled, we divide the analysis into three cases.

**Case 1)**  $t = t_2, q = m$ . In this case, all the available machines are already reserved, therefore it is infeasible to assign job  $j$  to time slot  $t_2$ .

**Case 2)**  $t = t_2, 0 < q < m$ . In this case, job  $j$  is assigned to time slot  $t_2$ . Since  $q > 0$ , the calibration setting on time slot  $t_2$  is already determined, we only need to choose an unoccupied machine for job  $j$  at slot  $t_2$  and recurse to the sub-problem  $f(j-1, t_1, t_2, q+1, \mathbf{c}_{t_1}, \mathbf{c}_{t_2})$ .

**Case 3)** In this case, job  $j$  can be scheduled at time slot  $t$ , and time slot  $t$  is not yet active so far. We enumerate the calibrations that cover time slot  $t$ , i.e., vector  $\mathbf{c}_t \in \mathcal{C}(t)$ . As argued in Proposition 7, when job  $j$  is scheduled at time slot  $t$ , jobs  $J(j-1, t_1, t)$  and  $J(j-1, t, t_2)$  should be scheduled during intervals  $(t_1, t]$  and  $(t, t_2]$  respectively. Then We determine the calibrations that cross the boundary of the intervals. We define  $\dot{\mathbf{c}}_t = \max\{\iota(\mathbf{c}_{t_1}, t - t_1), \mathbf{c}_t\}$  to include all the calibrations in  $\mathbf{c}_{t_1}$  or  $\mathbf{c}_t$  and restrict  $\dot{\mathbf{c}}_t \in \mathcal{C}(t)$  to avoid the situation where the number of available machines exceeds  $m$ . We reserve one machine for the job  $j$  at time slot  $t$  in the first sub-problem  $f(j-1, t_1, t, 1, \mathbf{c}_{t_1}, \dot{\mathbf{c}}_t)$ . Symmetrically, we define vector  $\ddot{\mathbf{c}}_t = \max\{\iota(\mathbf{c}_{t_2}, t - t_2), \mathbf{c}_t\}$  and the second sub-problem to be  $f(j-1, t, t_2, q, \ddot{\mathbf{c}}_t, \mathbf{c}_{t_2})$ . As we have tried every possibility of time slot  $t$ , vector  $\mathbf{c}_t$ , at least one try is the same as the optimal solution.  $\blacktriangleleft$

**Time Complexity.** The dynamic program has table size  $O(n^2|\Phi|^2|\mathcal{C}|^2)$ . Constructing the solution from the sub-problems takes  $O(|\Phi||\mathcal{C}|)$  steps. In total, the running time is  $O(n^2|\Phi|^3|\mathcal{C}|^3) = O(n^8m^{3T})$ . When  $T$  is constant, our dynamic programming approach has polynomial running time.

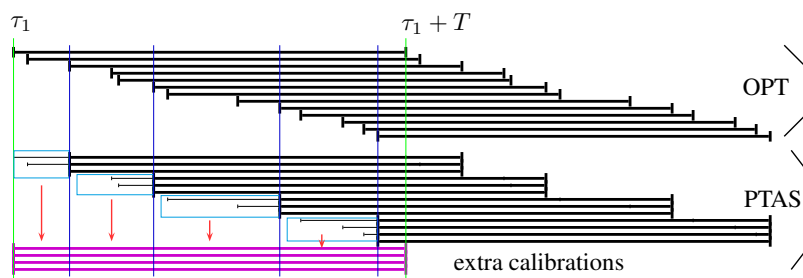
**Connection with Section 3.** When  $m$  is input, the dynamic programming approach in Section 3 has exponential running time, while the running time of the approach in this section is polynomial on  $n$  and  $m$ . However, this running time is exponential on  $T$ . When  $T$  is constant, our proposed dynamic programming approach has polynomial running time.

## 5 PTAS

In this section, we extend the dynamic programming approach in Section 3 and present a PTAS. In other words, for any constant  $\epsilon > 0$ , we give a  $(1 + \epsilon)$ -approximation solution with  $m$  machines. The high level idea of the PTAS is to compress the vectors by decreasing the number of possible distinct starting times of calibrations so that the dynamic programming



has polynomial running time. We propose a compression method by delaying the calibrations in the optimal solution and prove that in the approximation solution, for each time slot  $t$  the corresponding vector set  $\Gamma(t)$  has polynomial cardinality. More specifically, we prove that given  $(1 + \epsilon)m$  machines there exists a  $(1 + \epsilon)$ -approximation solution such that for any time slot  $t$ , the number of distinct starting times (and ending times) of the calibrations that cover time slot  $t$  is at most  $2\lceil 1/\epsilon \rceil + 1$ . At last, we use a modified dynamic programming algorithm to find that  $(1 + \epsilon)$ -approximation solution without using the extra  $\epsilon m$  machines.



■ **Figure 3** An illustration for Lemma 11 to show the transformation of the calibrations in the optimal solution. For each group of calibrations, after delaying the calibrations the affected jobs are depicted within a rectangle. We move the affected jobs in each rectangle into the extra machines, without changing the job starting time. As the time interval covered by each rectangle is disjoint with other rectangles, the new schedule of the jobs from two rectangles is independent of each other.

► **Lemma 11.** *There exists a  $(1 + \epsilon)$ -approximation solution on  $(1 + \epsilon)m$  machines such that for any time slot  $t$ , the number of distinct starting times (and ending times) of the calibrations that cover time slot  $t$  is at most  $2\lceil 1/\epsilon \rceil + 1$ , and there are at most  $m$  jobs scheduled at time slot  $t$ .*

**Proof.** Consider the optimal solution that satisfies Lemma 1 and Lemma 3, in which no two calibrations overlap with each other on the same machine. We show how to transform the optimal solution into another solution satisfying the statement, shown in Figure 3. In our approach, we maintain the schedule of the jobs as in the optimal schedule and only change the schedule of calibrations. Hence, in the new schedule there are at most  $m$  jobs scheduled at any time slot. Assume that in the optimal solution the calibrations are sorted in non-decreasing order of their starting times (regardless of the machines). We define *block* to be the set of consecutive calibrations satisfying the property that the largest difference of their starting times is less than  $T$  and the set is maximal in the sense that adding one more calibration will violate the property. We partition the calibrations in the optimal solution into many disjoint blocks starting from the first calibration. The transformation works in many phases where in each phase we work on one block.

Suppose in the current phase the block contains  $l$  calibrations. First, we will delay these  $l$  calibrations so that the number of distinct starting times of these calibrations is at most  $\lceil 1/\epsilon \rceil$  (skip the phase and do nothing if  $l \leq \lceil 1/\epsilon \rceil$ ). This process will cause the infeasibility of some jobs because of the delay of calibrations. We construct a feasible schedule of jobs by creating an extra  $\lceil \epsilon l \rceil$  calibrations on the extra  $\epsilon m$  machines and rescheduling the affected jobs on the extra machines.

Let  $\tau_i$  be the starting time of the  $i$ -th calibration in the block, we have  $\tau_1 \leq \tau_2 \leq \dots \leq \tau_l < \tau_1 + T$ . Note that  $l \leq m$  because each of these calibrations covers time slot  $\tau_1 + T$ .

## 20:10 Revisit the Scheduling Problem with Calibrations

**Step 1. (Partition)** We partition the calibrations in the block into  $\lceil 1/\epsilon \rceil$  groups such that each group contains at most  $\lceil \epsilon l \rceil$  consecutive calibrations. One would find that the partition is feasible as  $\lceil 1/\epsilon \rceil \cdot \lceil \epsilon l \rceil \geq l$ . Note that the maximum calibration starting time in one group is no larger than the minimum calibration starting time in the next group.

**Step 2. (Delay)** For each group of calibrations, let  $\tau$  be the latest calibration starting time, then we delay the calibrations in this group so that they have identical starting time  $\tau$ .

**Step 3. (Augment)** We add an extra group of  $\lceil \epsilon l \rceil$  calibrations once with identical starting time  $\tau_1$  on the extra  $\epsilon m$  machines.

**Step 4. (Transform)** For the calibrations that are delayed in each group, we reschedule the corresponding affected jobs on the extra machines, without changing the starting time of any job.

**Analysis.** We show that the new schedule of jobs is feasible. First, we claim that the number of the affected jobs which start at a time  $t \in [\tau_1, \tau_1 + T)$  in the optimal solution is at most  $\lceil \epsilon l \rceil - 1$ . In total we have created an extra of  $\lceil \epsilon l \rceil$  calibrations. For each group of calibrations, let  $t_a, t_b$  be the smallest and largest calibration starting time, respectively. There is at least one calibration that is not delayed, hence we delay at most  $\lceil \epsilon l \rceil - 1$  calibrations in this group. The affected jobs caused by the delay of the calibrations in this group must have job starting time within  $[t_a, t_b)$ . In other words, in the optimal solution any affected job that starts at a time within  $[t_a, t_b)$  must be scheduled in a calibration from this group (i.e., not from other groups), because the maximum calibration starting time in one group is no larger than the minimum calibration starting time in the next group by Step 1. Therefore,  $\forall t \in [\tau_1, \tau_1 + T)$ , the total number of the affected jobs which start at time  $t$  is at most  $\lceil \epsilon l \rceil - 1$ , because we delay at most  $\lceil \epsilon l \rceil - 1$  calibrations in this group. Since  $\lceil \epsilon l \rceil - 1 \leq \lceil \epsilon l \rceil$ , we conclude that the new schedule of jobs is feasible.

Now, we prove the lemma. Suppose in total there are  $b$  phases and let  $\tau'_1, \tau'_2, \dots, \tau'_b$  be the starting times of the extra calibrations in each phase. Then we have  $\tau'_{i+1} - \tau'_i \geq T$ ,  $\forall i \in [1, b)$  according to the above process. In one phase, we process  $l$  calibrations and we create  $\lceil \epsilon l \rceil$  extra calibrations, which implies that the final solution is  $(1 + \epsilon)$ -approximation since we never consider a calibration twice in different phases. Moreover Lemma 3 holds for the new solution because we do not change the starting time of any job and the starting time of the extra calibrations in each phase is the same as one of the calibrations from the block in the optimal solution. And note that in the new solution, calibrations might overlap with each other on the same machines. In each phase, we partition the calibrations into  $\lceil 1/\epsilon \rceil$  groups and the calibrations in each group have identical starting time, hence the number of distinct starting times of the calibrations is at most  $\lceil 1/\epsilon \rceil + 1$  in each phase, including the extra calibrations. In other words, for each interval  $[\tau'_i, \tau'_{i+1})$  the number of distinct starting times of the calibrations that start during this interval in the new solution is at most  $\lceil 1/\epsilon \rceil + 1$ . Consider an arbitrary time slot  $t$  from  $[\tau'_i, \tau'_{i+1})$  and the calibrations that cover slot  $t$  in the new solution. These calibrations must have starting times in  $(\tau'_{i-1}, \tau'_{i+1})$  since  $\tau'_{i+1} - \tau'_{i-1} \geq 2T$ . Therefore, the total number of distinct starting times of the calibrations that cover time slot  $t$  is at most  $2\lceil 1/\epsilon \rceil + 1$  (the extra calibrations in phase  $i - 1$  will not cover time slot  $t$ ). Also, the total number of distinct ending times of the calibrations that cover time slot  $t$  is at most  $2\lceil 1/\epsilon \rceil + 1$ , because all calibrations have identical length. Eventually, the lemma is proved.  $\blacktriangleleft$

### Vector Compression

Lemma 11 shows that for each optimal solution, there is a corresponding  $(1+\epsilon)$ -approximation solution on  $(m + \epsilon m)$  machines with the property that the number of distinct starting times of the calibrations that cover each time slot  $t$  is bounded by a constant. In the following, we propose a method to find a  $(1 + \epsilon)$ -approximation solution with  $m$  machines, which is based on the dynamic programming in the previous section. We first propose the algorithm with  $(m + \epsilon m)$  machines (which we refer to as *resource-augmentation* version), while guaranteeing that for any time slot there are at most  $m$  jobs scheduled. Note that the extra  $\epsilon m$  machines is due to the extra additional calibrations as shown in Figure 3. Therefore, we then transform the resource-augmentation solution to a solution that only requires  $m$  machines by rescheduling the calibration without changing the schedule of jobs.

Let  $h = 2\lceil 1/\epsilon \rceil + 1$ ,  $m' = m + \lfloor \epsilon m \rfloor$  and we assume  $m' < n$  (a trivial solution could be found when the optimal schedule uses  $m'$  machines with  $m' \geq n$ ). Previously in Section 3, for the calibrations that cover time slot  $t$ , we use vector  $\boldsymbol{\gamma} = \langle \gamma_1, \gamma_2, \dots, \gamma_m \rangle \in \Gamma(t)$  to mark the calibration starting time on each machine. Similar to Section 4, in the modified dynamic programming for PTAS, we discard the information of the mapping from calibrations to machines and only mark the starting times of the calibrations that cover time slot  $t$ . In other words, for each calibration, we do not need to know the corresponding machine on which the calibration takes effect. We focus on the solution on  $m'$  machines and use *configurations* to mark the calibration starting times.

► **Definition 12.** We define a configuration to be a pair of vectors  $\langle \boldsymbol{\alpha}, \boldsymbol{\eta} \rangle$  where  $\boldsymbol{\alpha} = \langle \alpha_1, \alpha_2, \dots, \alpha_h \rangle$ ,  $\boldsymbol{\eta} = \langle \eta_1, \eta_2, \dots, \eta_h \rangle$  and for each  $i \in [1, h]$ ,  $\alpha_i \in \{NUL\} \cup \Psi$  indicates the starting time of a calibration,  $\eta_i \in [0, m']$  indicates the number of the calibrations that share the same starting time  $\alpha_i$ . We define  $\mathcal{A} = \{ \langle \alpha_1, \alpha_2, \dots, \alpha_h \rangle \mid \alpha_i \in \{NUL\} \cup \Psi, \forall i \in [1, h] \}$  to be the set of all possible vectors  $\boldsymbol{\alpha}$ . Given time slot  $t$ , we define  $\mathcal{A}(t) = \{ \langle \alpha_1, \alpha_2, \dots, \alpha_h \rangle \mid \alpha_i \in \{NUL\} \cup (\Psi \cap [t - T, t]), \forall i \in [1, h] \}$ , where  $\alpha_i$  indicates the starting time of a calibration which covers time slot  $t$ . Let  $\mathcal{B} = \{ \boldsymbol{\eta} \mid \sum_{i=1}^h \eta_i \leq m', \eta_i \in [0, m'], \forall i \in [1, h] \}$  be the set of all possible vectors  $\boldsymbol{\eta}$ .

In total we have at most  $n^h |\Psi|^h$  configurations as  $|\mathcal{A}| \leq |\Psi|^h$  and  $|\mathcal{B}| \leq (m' + 1)^h \leq n^h$ , which implies that the total number of possible configurations is polynomial in  $n$ . Given time slot  $t$ , and two configurations  $\langle \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\eta}} \rangle$  and  $\langle \check{\boldsymbol{\alpha}}, \check{\boldsymbol{\eta}} \rangle$ , we use  $\cup_t$  as the notation of the process that merges these two configurations into a new configuration  $\langle \boldsymbol{\alpha}, \boldsymbol{\eta} \rangle$  where  $\langle \boldsymbol{\alpha}, \boldsymbol{\eta} \rangle = \langle \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\eta}} \rangle \cup_t \langle \check{\boldsymbol{\alpha}}, \check{\boldsymbol{\eta}} \rangle$  such that each calibration from the new configuration covers time slot  $t$ . In the merging process, we first identify the distinct starting times of the calibrations from the two configurations that cover time slot  $t$ , then for each distinct starting time, we count the number of calibrations that share the same starting time. We guarantee that  $\boldsymbol{\alpha} \in \mathcal{A}(t)$  and  $\boldsymbol{\eta} \in \mathcal{B}$  for the new configuration  $\langle \boldsymbol{\alpha}, \boldsymbol{\eta} \rangle$  after the merging process. In other words, we will discard the new configuration if it is not valid (either the number of distinct calibration starting times is beyond  $h$  or  $\boldsymbol{\eta} \notin \mathcal{B}$ ).

► **Definition 13.** We define  $f^\#(j, t_1, t_2, q, \langle \check{\boldsymbol{\alpha}}, \check{\boldsymbol{\eta}} \rangle, \langle \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\eta}} \rangle)$  to be the minimum number of extra necessary calibrations to schedule jobs  $J(j, t_1, t_2)$  on  $m'$  machines where  $j \in J, t_1 \in \Phi, t_2 \in \Phi, q \in [0, m], \check{\boldsymbol{\alpha}} \in \mathcal{A}(t_1), \hat{\boldsymbol{\alpha}} \in \mathcal{A}(t_2), \check{\boldsymbol{\eta}} \in \mathcal{B}, \hat{\boldsymbol{\eta}} \in \mathcal{B}$  on the condition that

- i.) jobs in  $J(j, t_1, t_2)$  are only scheduled during time interval  $(t_1, t_2]$ .
- ii.) calibrations indicated by configurations  $\langle \check{\boldsymbol{\alpha}}, \check{\boldsymbol{\eta}} \rangle$  and  $\langle \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\eta}} \rangle$  have already been selected to be assigned to machines.
- iii.)  $q$  other jobs (not from  $J(j, t_1, t_2)$ ) have already been assigned at time slot  $t_2$ .
- iv.) there are at most  $m$  jobs scheduled at any time slot.

## 20:12 Revisit the Scheduling Problem with Calibrations

In dynamic programming, we allow the overlap of calibrations and we guarantee that at each time slot, we assign at most  $m$  jobs into this slot.

► **Proposition 14.** *Let  $F^\# = f^\#(j, t_1, t_2, q, \langle \check{\alpha}, \check{\eta} \rangle, \langle \hat{\alpha}, \hat{\eta} \rangle)$ . For the base case  $J(j, t_1, t_2) = \emptyset$ , we set  $F^\#$  to be 0 if time slot  $t_2$  is covered by at least  $q$  calibrations given by configurations  $\langle \check{\alpha}, \check{\eta} \rangle$  and  $\langle \hat{\alpha}, \hat{\eta} \rangle$ , otherwise  $\infty$ . If  $j \notin J(j, t_1, t_2)$ , we have  $F^\# = f^\#(j - 1, t_1, t_2, q, \langle \check{\alpha}, \check{\eta} \rangle, \langle \hat{\alpha}, \hat{\eta} \rangle)$ . If  $j \in J(j, t_1, t_2)$  and  $\Phi(j) \cap (t_1, t_2] = \emptyset$ , we have  $F^\# = \infty$ . Otherwise, we have  $F^\# =$*

$$\min_{t \in \Phi(j) \cap (t_1, t_2]} \begin{cases} \infty & , \text{ if } t = t_2, q = m \\ f^\#(j - 1, t_1, t_2, q + 1, \langle \check{\alpha}, \check{\eta} \rangle, \langle \hat{\alpha}, \hat{\eta} \rangle) & , \text{ if } t = t_2, 0 < q < m \\ \min_{\text{cond.}} \sum_{i=0}^h \eta_i \\ + f^\#(j - 1, t_1, t, 1, \langle \check{\alpha}, \check{\eta} \rangle, \langle \hat{\alpha}, \hat{\eta} \rangle) \\ + f^\#(j - 1, t, t_2, q, \langle \check{\alpha}, \check{\eta} \rangle, \langle \hat{\alpha}, \hat{\eta} \rangle) & , \text{ if } t < t_2 \text{ or } q = 0 \end{cases}$$

where *cond.* stands for  $\langle \check{\alpha}, \check{\eta} \rangle = \langle \alpha, \eta \rangle \cup_t \langle \hat{\alpha}, \hat{\eta} \rangle$ ,  $\langle \check{\alpha}, \check{\eta} \rangle = \langle \alpha, \eta \rangle \cup_t \langle \check{\alpha}, \check{\eta} \rangle$  where  $\alpha, \hat{\alpha}, \check{\alpha} \in \mathcal{A}(t)$  and  $\eta, \hat{\eta}, \check{\eta} \in \mathcal{B}$ .

**Proof.** Similar to Proposition 7, we maintain the invariant that whenever a time slot becomes active (i.e., we reserve a time slot for a job), we enumerate all the calibrations that cover this time slot (excluding the calibrations that have already been selected). For example, for time slot  $t$  which is not active at the moment, even some calibrations from configurations  $\langle \check{\alpha}, \check{\eta} \rangle, \langle \hat{\alpha}, \hat{\eta} \rangle$  might already cover time slot  $t$ . We enumerate the remaining calibrations that could cover time slot  $t$  (i.e., configuration  $\langle \alpha, \eta \rangle$  with  $\alpha \in \mathcal{A}(t), \eta \in \mathcal{B}$ ) when we plan to assign a job to time slot  $t$ . Note that the actual calibrations that cover time slot  $t$  come from configurations  $\langle \check{\alpha}, \check{\eta} \rangle, \langle \hat{\alpha}, \hat{\eta} \rangle$  and  $\langle \alpha, \eta \rangle$ . In the dynamic programming, we enumerate the time slot  $t$  in which job  $j \in J(j, t_1, t_2)$  is scheduled in the optimal schedule.

**Case 1)** If  $t = t_2, q = m$ , we cannot assign job  $j$  to time slot  $t$  because there are already other  $m$  jobs assigned to time slot  $t$ .

**Case 2)**  $t = t_2, 0 < q < m$ . In this case, time slot  $t$  has already become active since  $q > 0$ . Hence, the calibrations that cover time slot  $t_2$  has already been enumerated (in other words, we do not need to enumerate it again). Also, job  $j$  could be assigned to time slot  $t_2$  because  $q < m$ . Therefore, we just assign job  $j$  to time slot  $t$  and reduce to the sub-problem  $f^\#(j - 1, t_1, t_2, q + 1, \langle \check{\alpha}, \check{\eta} \rangle, \langle \hat{\alpha}, \hat{\eta} \rangle)$ . Especially, if  $q = 0$ , no job has been assigned to time slot  $t$ , i.e., time slot  $t$  is not active, then we have to enumerate the calibrations that cover time slot  $t$ , which is handled in Case 3). If  $t < t_2$ , time slot  $t$  is not active because we only reserve time slot  $t_2$  for the  $q$  other jobs.

**Case 3)** In this case, job  $j$  can be scheduled at time slot  $t$ , and no job has been assigned to time slot  $t$  so far. We enumerate the calibrations that cover time slot  $t$ , which is the configuration  $\langle \alpha, \eta \rangle$  with  $\alpha \in \mathcal{A}(t), \eta \in \mathcal{B}$ . As argued in Proposition 7, when job  $j$  is scheduled at time slot  $t$ , jobs  $J(j - 1, t_1, t)$  and  $J(j - 1, t, t_2)$  should be scheduled during intervals  $(t_1, t]$  and  $(t, t_2]$  respectively. Hence we reduce to sub-problems  $f^\#(j - 1, t_1, t, 1, \langle \check{\alpha}, \check{\eta} \rangle, \langle \hat{\alpha}, \hat{\eta} \rangle)$  and  $f^\#(j - 1, t, t_2, q, \langle \check{\alpha}, \check{\eta} \rangle, \langle \hat{\alpha}, \hat{\eta} \rangle)$  where  $\langle \check{\alpha}, \check{\eta} \rangle = \langle \alpha, \eta \rangle \cup_t \langle \hat{\alpha}, \hat{\eta} \rangle$ ,  $\langle \check{\alpha}, \check{\eta} \rangle = \langle \alpha, \eta \rangle \cup_t \langle \check{\alpha}, \check{\eta} \rangle$  by reserving a calibrated machine at time slot  $t$  for job  $j$ . Because we enumerate all possible configurations  $\langle \alpha, \eta \rangle$  with  $\alpha \in \mathcal{A}(t), \eta \in \mathcal{B}$ , we are able to reach the schedule that is the same as the optimal schedule. ◀

**Time complexity.** The modified dynamic program has table size  $O(n^2 |\Phi|^2 (|\mathcal{A}| |\mathcal{B}|)^2)$ . Constructing the solution from the sub-problems takes  $O(|\Phi| |\mathcal{A}| |\mathcal{B}|)$  steps. In total the time complexity is  $O(n^2 |\Phi|^3 (|\mathcal{A}| |\mathcal{B}|)^3) = O(n^2 |\Phi|^3 n^{3h} |\Psi|^{3h}) = O(n^{17+18\lceil 1/\epsilon \rceil})$ .

► **Lemma 15.** *There exists a  $(1 + \epsilon)$  - approximation solution on  $m$  machines.*

**Proof.** By Lemma 11 and the algorithm in Proposition 14, we can obtain a  $(1 + \epsilon)$  - approximation solution  $\sigma$  on  $m'$  machines while guaranteeing that there are at most  $m$  jobs scheduled at any time slot. In the following, we transform  $\sigma$  to an  $(1 + \epsilon)$  - approximation solution  $\sigma'$  on  $m$  machines, without changing the schedule of jobs. In solution  $\sigma$ , we consider the earliest time slot  $t$  such that more than  $m$  machines are calibrated at time slot  $t$ .

**Case 1)** If no such time slot  $t$  exists, we then assign the calibrations to  $m$  machines via Round-Robin method, as proposed in [3], and obtain a feasible solution on  $m$  machines.

**Case 2)** Otherwise, there must be some calibration starting at time  $t - 1$ , we then delay this calibration by one more time slot. Note that jobs are still feasible since there are at most  $m$  jobs scheduled at time slot  $t$ . We repeat the above process until Case 1) occurs and Case 1) eventually will occur since the sum of the calibration starting time is increasing after each delay operation. Thus, we finish the proof. ◀

**Connection with Section 4.** Note that in Section 3 we directly record the calibration times on each machine in the dynamic program, while in Section 4 and this section we propose different methods for vector compression to ensure a polynomial space of the proposed dynamic programs. Specifically, in Section 4, for a fixed time slot  $t$ , the number of possible distinct calibration starting times within interval  $[t, t + T)$  is constant since  $T$  is constant, instead, in this section when  $T$  is input, we apply the calibration delaying operation to ensure the polynomial number of distinct calibration starting times within interval  $[t, t + T)$ .

## 6 Conclusion

We study the scheduling problem with calibrations on multiple machines where we consider the schedule of unit-time processing jobs with release times and deadlines such that the total number of calibrations is minimized. We propose two dynamic programming approaches to solve the problem with running time  $O(n^{8+6m})$  and  $O(n^8 m^{3T})$  respectively. Thus when  $m$  is constant or  $T$  is constant, we can give an algorithm of polynomial running time. Moreover, we present a PTAS, which has running time  $O(n^{17+18\lceil 1/\epsilon \rceil})$ . This approach very likely works only on the case that the jobs have identical processing time. It would be worth challenging to tackle the open problem proposed by Bender et al. [3] about the complexity status on multiple machines with jobs of unit processing times.

---

### References

- 1 Eric Angel, Evripidis Bampis, Vincent Chau, and Vassilis Zissimopoulos. On the complexity of minimizing the total calibration cost. In *International Workshop on Frontiers in Algorithmics*, pages 1–12. Springer, 2017. doi:10.1007/978-3-319-59605-1\_1.
- 2 Eric Angel, Evripidis Bampis, Vincent Chau, and Vassilis Zissimopoulos. Calibrations scheduling with arbitrary lengths and activation length. *Journal of Scheduling*, 24(5):459–467, 2021. doi:10.1007/S10951-021-00688-5.
- 3 Michael A. Bender, David P. Bunde, Vitus J. Leung, Samuel McCauley, and Cynthia A. Phillips. Efficient scheduling to minimize calibrations. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '13, pages 280–287, New York, NY, USA, 2013. ACM. doi:10.1145/2486159.2486193.
- 4 B Bringmann, A Küng, and W Knapp. A measuring artefact for true 3d machine testing and calibration. *CIRP Annals-Manufacturing Technology*, 54(1):471–474, 2005.
- 5 Peter Brucker and P Brucker. *Scheduling algorithms*, volume 3. Springer, Heidelberg, 2007.

- 6 Chris Burroughs. New integrated stockpile evaluation program to better ensure weapons stockpile safety, security, reliability, 2006. URL: <http://www.sandia.gov/LabNews/060331.html>.
- 7 Vincent Chau, Shengzhong Feng, Minming Li, Yinling Wang, Guochuan Zhang, and Yong Zhang. Weighted throughput maximization with calibrations. In *Algorithms and Data Structures: 16th International Symposium, WADS 2019, Edmonton, AB, Canada, August 5–7, 2019, Proceedings 16*, pages 311–324. Springer, 2019. doi:10.1007/978-3-030-24766-9\_23.
- 8 Vincent Chau, Minming Li, Samuel McCauley, and Kai Wang. Minimizing total weighted flow time with calibrations. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '17, pages 67–76, New York, NY, USA, 2017. ACM. doi:10.1145/3087556.3087573.
- 9 Vincent Chau, Minming Li, Elaine Yinling Wang, Ruilong Zhang, and Yingchao Zhao. Minimizing the cost of batch calibrations. *Theoretical Computer Science*, 828:55–64, 2020. doi:10.1016/J.TCS.2020.04.020.
- 10 Hua Chen, Vincent Chau, Lin Chen, and Guochuan Zhang. Scheduling many types of calibrations. In *International Conference on Algorithmic Applications in Management*, pages 286–297. Springer, 2020. doi:10.1007/978-3-030-57602-8\_26.
- 11 Zuzhi Chen and Jialin Zhang. Online scheduling of time-critical tasks to minimize the number of calibrations. *Theoretical Computer Science*, 914:1–13, 2022. doi:10.1016/J.TCS.2022.01.040.
- 12 Jeremy T. Fineman and Brendan Sheridan. Scheduling non-unit jobs to minimize calibrations. In *Proceedings of the 27th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '15, pages 161–170, New York, NY, USA, 2015. ACM. doi:10.1145/2755573.2755605.
- 13 Ronald L Graham, Eugene L Lawler, Jan Karel Lenstra, and AHG Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, 5:287–326, 1979.
- 14 Oscar H. Ibarra and Chul E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM*, 22(4):463–468, October 1975. doi:10.1145/321906.321909.
- 15 Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *J. ACM*, 47(4):617–643, July 2000. doi:10.1145/347476.347479.
- 16 Kuo-Huang Lin and Bin-Da Liu. A gray system modeling approach to the prediction of calibration intervals. *IEEE Transactions on Instrumentation and Measurement*, 54(1):297–304, 2005. doi:10.1109/TIM.2004.840234.
- 17 Hoai-Nhan Nguyen, Jian Zhou, and Hee-Jun Kang. A new full pose measurement method for robot calibration. *Sensors*, 13(7):9132–9147, 2013. doi:10.3390/S130709132.
- 18 Emilia Nunzi, Gianna Panfilo, Patrizia Tavella, Paolo Carbone, and Dario Petri. Stochastic and reactive methods for the determination of optimal calibration intervals. *IEEE Transactions on Instrumentation and Measurement*, 54(4):1565–1569, 2005. doi:10.1109/TIM.2005.851501.
- 19 Cynthia A Phillips, Cliff Stein, Eric Torng, and Joel Wein. Optimal time-critical scheduling via resource augmentation. In *Proceedings of the 29th. ACM Symposium on Theory of Computing (STOC '97)*, pages 140–149, New York, NY, 1997. ACM Press.
- 20 SR Postlethwaite, DG Ford, and D Morton. Dynamic calibration of CNC machine tools. *International Journal of Machine Tools and Manufacture*, 37(3):287–294, 1997.
- 21 Sartaj K. Sahni. Algorithms for scheduling independent tasks. *J. ACM*, 23(1):116–127, January 1976. doi:10.1145/321921.321934.
- 22 Petra Schuurman and Gerhard J. Woeginger. Approximation schemes – a tutorial, 2007.
- 23 Minze Stuiver, Paula J Reimer, and Thomas F Braziunas. High-precision radiocarbon age calibration for terrestrial and marine samples. *Radiocarbon*, 40(3):1127–1151, 1998.
- 24 Kai Wang. Calibration scheduling with time slot cost. *Theoretical Computer Science*, 821:1–14, 2020. doi:10.1016/J.TCS.2020.03.018.
- 25 Gerhard J Woeginger. When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? *INFORMS Journal on Computing*, 12(1):57–74, 2000. doi:10.1287/IJOC.12.1.57.11901.

- 26 G Zhang and R Hocken. Improving the accuracy of angle measurement in machine calibration. *CIRP Annals-Manufacturing Technology*, 35(1):369–372, 1986.
- 27 Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000. doi:10.1109/34.888718.