Revisit the Scheduling Problem with Calibrations

Yixiong Gao (City University of Hong Kong)
Joint work with
Lin Chen (Zhejiang University)
Minming Li (City University of Hong Kong)
Guohui Lin (University of Alberta)
Kai Wang (City University of Hong Kong)

ISAAC 2024

Outline

- Introduction
- Dynamic Programming Approach
- PTAS
- Conclusion

Motivation & Application

- Initiated from the Integrated Stockpile Evaluation (ISE) program which tests nuclear weapons periodically.

- Extend to the model where the machines need to be calibrated periodically to ensure high-quality products
  - Robotics
  - Digital cameras

    …

Formulation

- A set J of n jobs, where each job $j \in J$ has
  - a release time r_j
  - a deadline d_j
  - a processing time $p_j = 1$

- m identical parallel machines
  - a machine must be calibrated before it runs a job
  - calibrating a machine is instantaneous in the ideal model
  - the calibration remains valid for a time period of length T

Objective

- Minimize the number of calibrations

  (ensuring completes all jobs before their deadlines)

- A feasible solution includes:
  - Schedule of calibrations (when to start a calibration)
  - Schedule of jobs (when and on which machine to start a job)

Results Summary

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

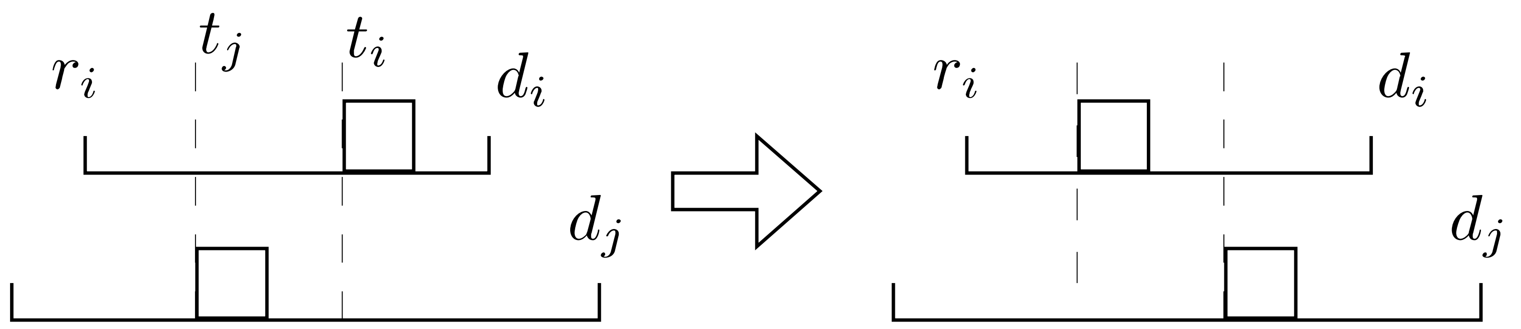
Earliest-Deadline-First Scheduling

- Sort the jobs by non-decreasing order of their deadlines, and non-decreasing order of release times if they have the same deadline.
- Index jobs from 1 to n after sorting.
- Once the schedule of calibrations is fixed:
  - consider each time slot t, i.e. time interval $(t-1, t]$
  - the job with smallest index has the highest priority

Earliest-Deadline-First Scheduling

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)



The value $(i - t_i)^2 + \left(j - t_j\right)^2$ strictly decrease after swapping

$\Rightarrow$ Every optimal schedule can be transformed into another optimal solution following EDF after a finite number of the swapping process

Reduce the number of useful time slots

Defination 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*
$\Phi(j) = \{t \mid r_j < t \le d_j, t \in \Phi\}, \forall\, j \in J$.

$|\Psi| = O(n^2)$ and $|\Phi| = O(n^2)$

Lemma 3. *[1] There always 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* $\Phi(j)$.

[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

Outline

- Introduction
- Dynamic Programming Approach
  - when m is constant
  - when T is constant
- PTAS
- Conclusion

Vector Representation of Calibrations

Mark the calibrations on each machine that cover time slot t by

$$\boldsymbol{\gamma} = \langle \gamma_1, \gamma_2, \dots, \gamma_m \rangle$$

where $\gamma_k \in \{NUL\} \cup (\Psi \cap \ [t-T, t))$ indicates the starting time of the calibration on machine k.

$$\Gamma(t) = \ \{\langle \gamma_1, \gamma_2, \dots, \gamma_m \rangle \mid \gamma_k \in \{\text{NUL}\} \cup \left(\Psi \cap \ [t-T, t)\right), \forall\, k \in \ [1, m]\}$$

to be the set of all possible vectors, with respect to time slot t.

We define operator < (or $\leq, >, \geq$, analogously) to be $\beta = \ (\lambda < x)$ where $\beta_k = \lambda_k$ if $\lambda_k \neq \text{NUL}, \lambda_k < x$ and otherwise $\beta_k = \text{NUL}$.

Observation

Defination 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., $J(j, t_1, t_2) = \{i | i \leq j, r_i \in [t_1, t_2), i \in J\}$

If jobs $J(j, t_1, t_2)$ are only scheduled during $[t_1, t_2)$, and job j is scheduled at time slot t :

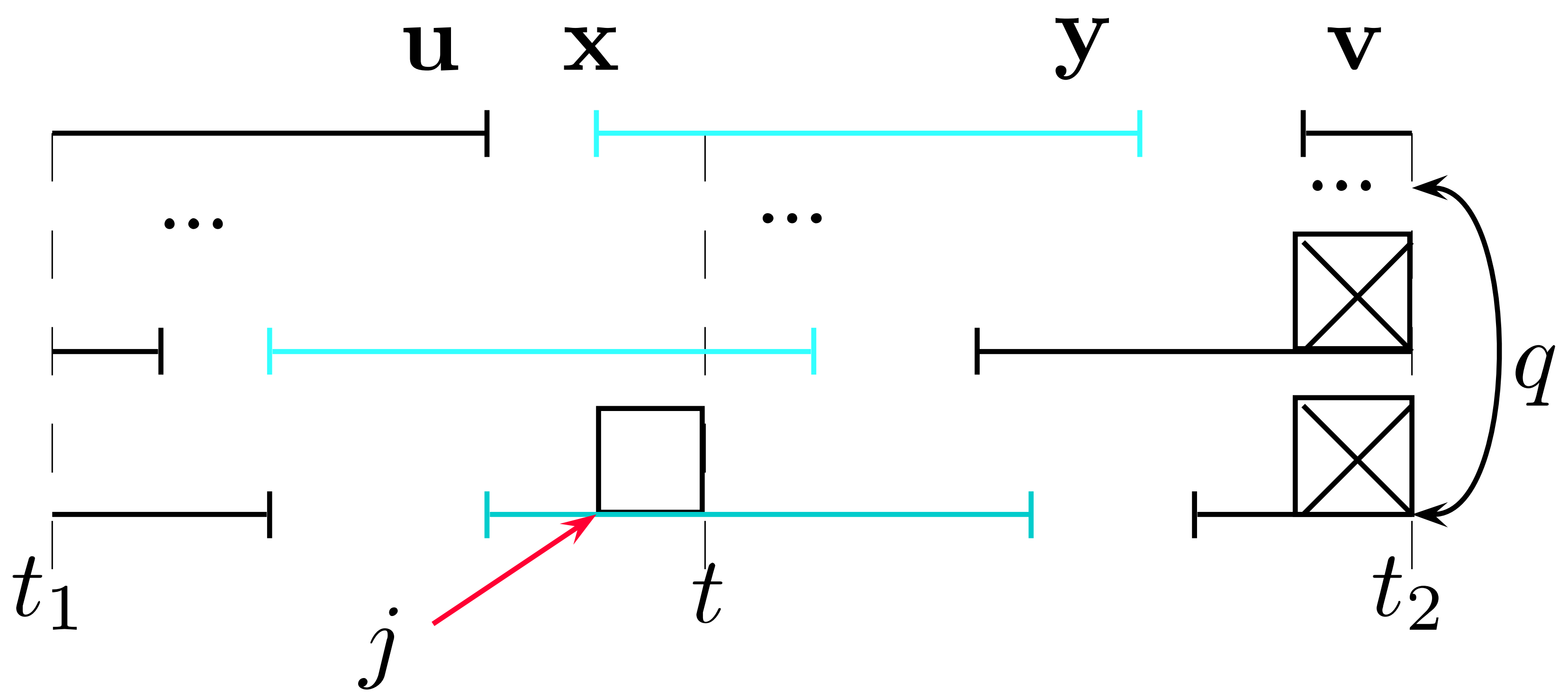
- $J(j-1, t_1, t)$ by EDF $\Rightarrow$ must be scheduled during $[t_1, t)$
- $J(j-1, t, t_2)$ by release time $\Rightarrow$ must be scheduled during $[t, t_2)$

Dynamic Programming States

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

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

Dynamic Programming Transitions



Dynamic Programming Transitions

Let $\boldsymbol{\delta}(\boldsymbol{\lambda}) = \sum_{\lambda_k \neq NUL, k \in [1,m]} 1$ on vector $\boldsymbol{\lambda}$.

Then $f(j, t_1, t_2, q, \boldsymbol{u}, \boldsymbol{v}) =$

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

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

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 $\boldsymbol{m}$ is constant, our dynamic programming approach has polynomial running time.

Improve the Vector Representation

We only need to distinguish calibrations by there starting time.

Mark the **numbers of calibrations** distinguished by the number of remaining time slots that cover time slot t by

$$c_t = \left\langle c_{t,1}, c_{t,2}, \cdots, c_{t,T} \right\rangle$$

where $c_{t,k} \in \{0,1,2,\ldots,m\}$ indicates the number of calibrations that makes the machine available at time slot $t, t+1, \ldots, t+k-1$.

Let $C(t) = \left\{ \left\langle c_{t,1}, c_{t,2}, \cdots, c_{t,T} \right\rangle \middle| c_{t,k} \in \{0,1,2,\ldots,m\}, \forall\, k \in [1,T] \wedge \sum_{k=1}^{T} c_{t,k} \leq m \right\}$ be the set of all possible vectors, with respect to time slot t.

Modified Dynamic Programming States

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

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

Dynamic Programming Transitions

$f(j, t_1, t_2, q, \boldsymbol{c_{t_1}}, \boldsymbol{c_{t_2}}) =$

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

Where $cond.$ stands for $\boldsymbol{\dot{c}_t} = \max\{\iota(c_{t_1}, t - t_1), c_t\}, \boldsymbol{\ddot{c}_t} = \max\{\iota(c_{t_2}, t - t_2), c_t\}$

where $\boldsymbol{c_t}, \boldsymbol{\dot{c}_t}, \boldsymbol{\ddot{c}_t} \in C(t)$ and $\iota(\boldsymbol{c_t}, x) = \boldsymbol{c_{t+x}}$ where $c_{\mathrm{t+x},k} = c_{t,k-x}$ if $k - x \in [1, T]$, and otherwise $0, \forall\ k \in [1, T]$.

Time Complexity

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

Outline

Ideas

- Compress the vectors by decreasing the number of possible distinct starting times of calibrations
  - By delaying the calibrations in the optimal solution

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*.*

- Maintain the schedule of the jobs as in the optimal schedule
- Only change the schedule of calibrations

Divide Calibrations into Blocks

- We define block to be the set of consecutive calibrations satisfying
  - the largest difference of their starting times is less than T
  - the set is maximal
- Partition the calibrations in the optimal solution into many disjoint blocks starting from the first calibration.

- Suppose a block contains l calibrations.
- Let τ_i be the starting time of the i-th calibration in the block, we have $\tau_1 \leq \tau_2 \leq \cdots \leq \tau_l < \tau_1 + T$.
- Also $l \leq m$ because each of these calibrations covers time slot $\tau_1 + T$.

Transformation in Blocks

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.

- the partition is feasible as $\lceil 1/\epsilon \rceil \times \lceil \epsilon l \rceil \geq l$
- the maximum calibration starting time in one group is no larger than the minimum calibration starting time in the next group.

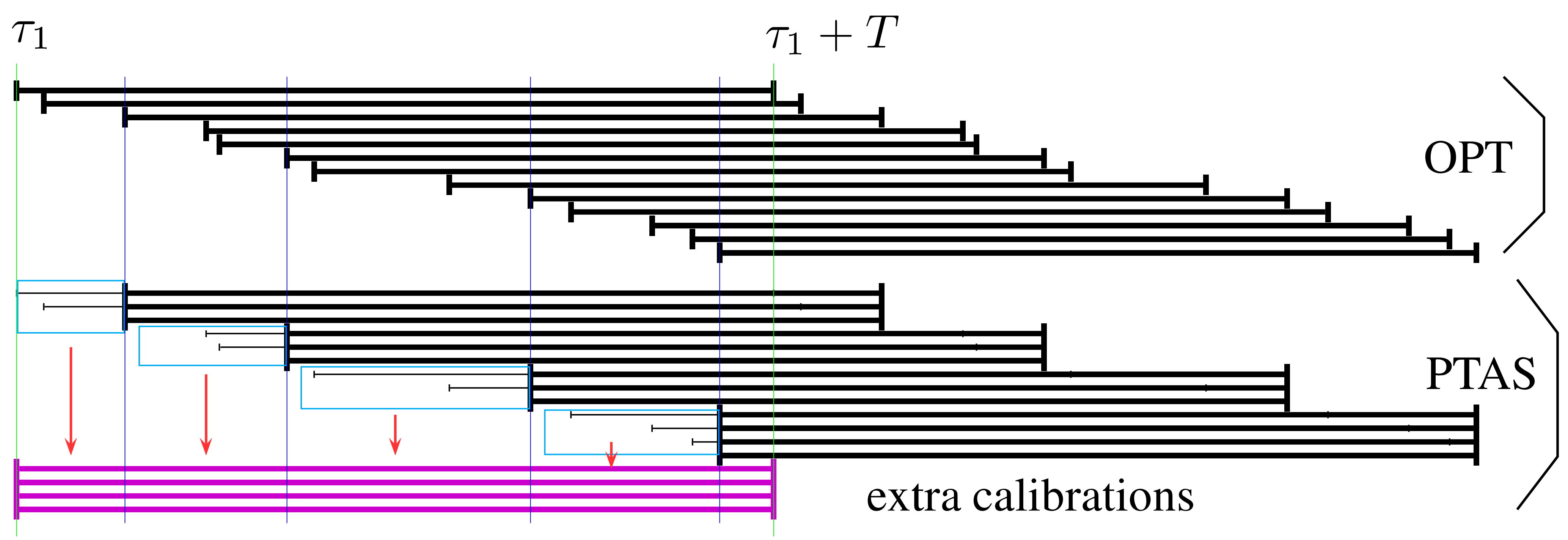
Step 2. **(Delay)** For each group of calibrations, let τ be the latest calibration starting time, then we delay the calibrations in this group so that they have identical starting time τ.

Transformation in Blocks

Step 3. **(Augment)** We add an extra group of $\lceil \epsilon l \rceil$ calibrations once with identical starting time τ_1 on the extra ϵm 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.

Transformation in Blocks



Vector Compression

- We define a **configuration** to be a pair of vectors $\langle \alpha, \eta \rangle$ where $\alpha = \langle \alpha_1, \alpha_2, \dots, \alpha_h \rangle, \eta = \langle \eta_1, \eta_2, \dots, \eta_h \rangle$ 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 α_i.
- We define $A = \{ \langle \alpha_1, \alpha_2, \dots, \alpha_h \rangle | \, \alpha_i \in \{\mathrm{NUL}\} \cup \Psi, \forall\, i \in [1, h]\}$ to be the set of all possible vectors α. Given time slot t, we define $A(t) = \{\langle \alpha_1, \alpha_2, \dots, \alpha_h \rangle | \, \alpha_i \in \{\mathrm{NUL}\} \cup (\Psi \cap [t - T, \mathrm{t})), \forall\, i \in [1, h]\}$
- Let $B = \{ \eta | \sum_{\mathrm{i}=0}^{h} \eta_i \leq m', \eta_i \in [0, m'], \forall\, i \in [1, h]\}$ be the set of all possible vectors η.

Dynamic Programming States

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

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

Dynamic Programming Transitions

$$f^{\#}(j, t_1, t_2, q, \langle \breve{\alpha}, \check{\eta} \rangle, \langle \hat{\alpha}, \hat{\eta} \rangle) =$$

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

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

Time Complexity

- The modified dynamic program has table size $O(n^2|\Phi|^2(|A||B|)^2)$.
- Constructing the solution from the sub-problems takes $O(|\Phi||A||B|)$ steps.
- In total the time complexity is

$$O(n^2|\Phi|^3(|A||B|)^3) = O(n^2\,|\Phi|^3 n^{3h}\,|\Psi|^{3h}) = O(n^{17+18\lceil\frac{1}{\epsilon}\rceil})$$

Remove the extra machines

- 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, 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. repeat the above process until Case 1) occurs.

Outline

- Introduction
- Dynamic Programming Approach
- PTAS
- Conclusion

Results Summary

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

Future Work

- Tackle the complexity status on multiple machines with jobs of unit processing times.

- Design algorithms for the jobs with non-unit processing time.

Thank you!

Questions?