Efficient Scheduling to Minimize Calibrations

Michael Bender (Stony Brook U., Tokutek)
David Bunde (Knox College)
Vitus Leung (Sandia National Laboratories)
Samuel McCauley (Stony Brook U.)
Cynthia A. Phillips (Sandia National Laboratories)
Integrated Stockpile Evaluation (ISE)

- Monitor nuclear weapons stockpile for any unforeseen problems
- Check some number of weapons each year
- Uses equipment that require calibrations
  - Verify correct measurements (like adjusting a scale)
- Calibrations don’t take long, but are expensive
- Calibrations last for a specific amount of time, regardless of use

Goal: Minimize the number of calibrations

http://goingforwardblog.wordpress.com/2013/11/04/the-sticking-point/
The ISE Problem

- N unit-length jobs with release dates and deadlines.
- P identical machines
- A machine can only run a job if it has been calibrated in the last T time steps (interval of length T)
- Objective: minimize number of calibrations
  - Calibration takes no time

Recalibrate

April 3, 2014

Aussois Scheduling Workshop
Results

- **Optimal algorithm for one machine**
  - If there is a feasible one-machine schedule, it is optimal
    - More machines cannot help in this case
- **2-approximation for multiple machines**
  - Not known to be NP-hard for unit-sized jobs
  - Weakly NP-hard for jobs with non-unit length
- Some structural information for optimal schedules
Earliest-Deadline First

- For P machines, unit-sized jobs with release dates and deadlines, EDF finds a feasible schedule if one exists.
Scheduling Into Intervals

• Given a set of legal (calibrated) intervals
• EDF still finds a feasible schedule respecting intervals, if possible
  – Create dummy jobs to fill all forbidden times
  – Can always swap jobs scheduled at the same time to get this structure
One-Machine Algorithm Overview

- Lay down intervals incrementally
- Do not allow jobs between intervals
- Test incremental feasibility with EDF

Jobs can go here
Interesting Property

- It may be useful to **delay** a job when it’s possible to run it.
  - Not true for most objectives such as min makespan or min average completion time.
- A **push** moves an interval one time unit later.
- Idea: Push intervals as late as possible.
How Late is the Latest Possible?

• Push until any further would be infeasible (EDF test)

• Feasible push: current restrictions are feasible

• Infeasible push: Some job misses a deadline

April 3, 2014
Aussois Scheduling Workshop
Lazy Binning Algorithm

While jobs are not all scheduled
   Put down an interval immediately adjacent to the last interval
   Or at t=0 if there are no intervals
   Do all feasible pushes
   • Proof of correctness by examining first deviation from optimal

Example with T=3:

Release Times    Deadlines

April 3, 2014
Aussois Scheduling Workshop
Lazy Binning Example
Lazy Binning Example

\[
\begin{array}{cccc}
\text{j}_1 & \text{j}_2 & \text{j}_3 & \text{j}_4 \\
\text{j}_5 & \text{j}_6 & & \\
\end{array}
\]
Lazy Binning Example
Lazy Binning Example
Lazy Binning Example

\[ j_1 \quad j_2 \quad j_3 \quad j_4 \quad j_5 \quad j_6 \]
Lazy Binning Example
Lazy Binning Example

\[ j_1 \quad j_2 \quad j_3 \quad j_4 \quad j_5 \quad j_6 \]
One-Machine Schedules

- If a job set can be feasibly scheduled on one machine, the one-machine lazy-binning schedule is optimal
- Proof hint: Given multi-machine schedule, move an interval onto first without increasing # of intervals
- Active region grows forward and/or backward:
  - Add interval at the start, and push (moving others as necessary)
More Machines Can Help Otherwise

- Feasible on 2 machines, optimal on 3 machines
Multiprocessor Schedule: Round Robin

- Given a set of intervals and P machines
  - Sort intervals by starting time
  - Schedule round robin
- Algorithm doesn’t have to assign intervals to processors

```
M_1  [---I_1---]  [---I_4---]  [---I_7---]
M_2  [---I_2---]  [---I_5---]  [---I_8---]
M_3  [---I_3---]  [---I_6---]
```
Multiprocessor Partial Schedule Feasibility Check

- EDF on intervals
- EDF with remaining jobs on all permitted slots on processor P
  - After last interval on P
  - No earlier than start of last interval
Multiprocessors: Push Case 1

- We don’t lose a job
- May or may not gain a job
- Always good!
Multiprocessors: Push Case 2

- We lose a job scheduled before push and do not gain a job
  - To be feasible, \( j_{out} \) has to be schedulable in a future interval starting \( t \) or later
- Only good if we can eventually gain a new job

\[ \begin{array}{c}
\text{Deadline} \\
\hline
\text{Time } t \\
\end{array} \]

\[ \begin{array}{ccccccc}
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\end{array} \]
Multiprocessors: Push Case 3

- A pre-push job is replaced (in intervals) by a job earlier in EDF order.
- Always good. The job pushed out is more flexible.
Multiprocessors: Push Case 4

- A job is pushed out as in Case 2. Gain a job later in EDF order.
- Sometimes good, sometimes bad. Difficult to characterize.
Case 2 and 4 Can Help

• Multiprocessor lazy binning with conservative pushes gives:

• Next push of first interval is case 2 ($j_{out}$ but no $j_{in}$).
Case 2 and Case 4 Can Help

- Next push is Case 4 ($j_{in}$ is after $j_{out}$ in EDF order)
Case 2 and 4 Can Help

- Three calibrations for greedy. Two after “bad” pushes
P-Processor Approximation

• Simply, if there is a chance doing “bad” pushes can help, double interval to “cover all options”

Approx Alg

No Pushes

Some Pushes (Opt)
P-processor Approximation Algorithm

- If last interval placed at $t$ (initially 0), put an interval at $t$
- Move forward till there are at most $P$ intervals at $t$
- Do all good pushes (Case 1 or 3)
- If further pushing would be (0 or more Case 2)(Case 4) then double the interval size (and pay 2).
  - When doing EDF to characterize push, don’t use times after the end of current interval
Approximation Bound

- We add at most one interval per optimal interval: 2-approx
  - Proof hints
    - uses special structure of Case 2 and Case 4 pushes.
    - There’s an optimal schedule created by pushes
    - That optimal schedule does all Case-1 and Case-3
    - It may do some Case-2 and Case-4, but never more than T

- If all deadlines are distinct, no Case 4 pushes: optimal.

- One machine version is same as lazy-binning.
Open Problems

- Is this unit-length problem NP-complete? Or polytime algorithm?
- Min average completion time (e.g.) subject to calibration budget
  - Earlier more robust to disruptions
- Heterogeneous machines
- Calibration takes time
- Different job lengths