Scheduling independent stochastic tasks under deadline and budget constraints

Louis-Claude Canon\textsuperscript{1,2}, Aurélie Kong Win Chang\textsuperscript{1}, Yves Robert\textsuperscript{1,3}, Frédéric Vivien\textsuperscript{1}

1. FEMTO-ST, Université de Bourgogne Franche-Comté, France
2. LIP, Université de Lyon (CNRS, ENS Lyon, Inria, UCB Lyon), France
3. ICL, University of Tennessee Knoxville, USA

JLESC Short Talk – April 15, 2019
The little scheduling problem (1/2)

- Independent tasks, IID execution times with distribution \( \mathcal{D} \)
- Platform: identical processors, unit speed, unit cost
- User: limited budget \( b \) and execution deadline \( d \)
- **Objective:** maximize expected number of tasks completed

- Motivation: imprecise computing
- Tasks with mandatory parts + data-dependent optional parts
  \( \Rightarrow \) maximize \# optional parts
Scheduling policy

- Decide how many processors to launch & stop at each second
- Processors interrupted when deadline or budget is exceeded

- Each task can be deleted at any instant before completion
  - time/budget spent until interruption: completely lost
  - interrupted tasks cannot be relaunched (non-preemptive)
Core instance

- One processor
- Unlimited budget, no deadline
- Discrete distribution:

<table>
<thead>
<tr>
<th>Probability</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1 = 0.1$</td>
<td>$w_1 = 3$</td>
</tr>
<tr>
<td>$p_2 = 0.7$</td>
<td>$w_2 = 5$</td>
</tr>
<tr>
<td>$p_3 = 0.2$</td>
<td>$w_3 = 6$</td>
</tr>
</tbody>
</table>

- Objective: maximize success rate per time/budget unit $\mathcal{R}$
Illustrating example

Never interrupt tasks: 4 tasks completed.
Interrupt tasks after $w_1$: 1 task completed.
Interrupt tasks after $w_2$: 4 tasks completed.
Illustrating example

Never interrupt tasks: 4 tasks completed.
Interrupt tasks after $w_1$: 1 task completed.
Interrupt tasks after $w_2$: 4 tasks completed.
Illustrating example

Never interrupt tasks: 4 tasks completed.
Interrupt tasks after $w_1$: 1 task completed.
Interrupt tasks after $w_2$: 4 tasks completed.
Illustrating example

Never interrupt tasks: 4 tasks completed.
Interrupt tasks after $w_1$: 1 task completed.
Interrupt tasks after $w_2$: 4 tasks completed.
Illustrating example

Never interrupt tasks: 4 tasks completed.
Interrupt tasks after $w_1$: 1 task completed.
Interrupt tasks after $w_2$: 4 tasks completed.
Illustrating example

Never interrupt tasks: 4 tasks completed.
Interrupt tasks after $w_1$: 1 task completed.
Interrupt tasks after $w_2$: 4 tasks completed.
Illustrating example

Never interrupt tasks: 4 tasks completed.
Interrupt tasks after \( w_1 \): 1 task completed.
Interrupt tasks after \( w_2 \): 4 tasks completed.
Illustrating example

Never interrupt tasks: 4 tasks completed.
Interrupt tasks after $w_1$: 1 task completed.
Interrupt tasks after $w_2$: 4 tasks completed.
Illustrating example

Never interrupt tasks: 4 tasks completed.
Interrupt tasks after $w_1$: 1 task completed.
Interrupt tasks after $w_2$: 4 tasks completed.
Illustrating example

Never interrupt tasks: 4 tasks completed.
Interrupt tasks after $w_1$: 1 task completed.
Interrupt tasks after $w_2$: 4 tasks completed.
Illustrating example

Never interrupt tasks: 4 tasks completed.
Interrupt tasks after $w_1$: 1 task completed.
Interrupt tasks after $w_2$: 4 tasks completed.
Illustrating example

Never interrupt tasks: 4 tasks completed.
Interrupt tasks after $w_1$: 1 task completed.
Interrupt tasks after $w_2$: 4 tasks completed.
Illustrating example

Never interrupt tasks: 4 tasks completed.
Interrupt tasks after $w_1$: 1 task completed.
Interrupt tasks after $w_2$: 4 tasks completed.
Illustrating example

Never interrupt tasks: 4 tasks completed.
Interrupt tasks after $w_1$: 1 task completed.
Interrupt tasks after $w_2$: 4 tasks completed.
Illustrating example

Never interrupt tasks: 4 tasks completed.
Interrupt tasks after \( w_1 \): 1 task completed.
Interrupt tasks after \( w_2 \): 4 tasks completed.
Never interrupt tasks: 4 tasks completed.
Interrupt tasks after $w_1$: 1 task completed.
Interrupt tasks after $w_2$: 4 tasks completed.
Illustrating example

Never interrupt tasks: 4 tasks completed.
Interrupt tasks after $w_1$: 1 task completed.
Interrupt tasks after $w_2$: 4 tasks completed.
**Illustrating example**

Never interrupt tasks: 4 tasks completed.
Interrupt tasks after $w_1$: 1 task completed.
Interrupt tasks after $w_2$: 4 tasks completed.
Illustrating example

Never interrupt tasks: 4 tasks completed.
Interrupt tasks after $w_1$: 1 task completed.
Interrupt tasks after $w_2$: 4 tasks completed.
Illustrating example

Never interrupt tasks: 4 tasks completed.
Interrupt tasks after $w_1$: 1 task completed.
Interrupt tasks after $w_2$: 4 tasks completed.
Scheduling strategies

<table>
<thead>
<tr>
<th>Probability</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1 = 0.1$</td>
<td>$w_1 = 3$</td>
</tr>
<tr>
<td>$p_2 = 0.7$</td>
<td>$w_2 = 5$</td>
</tr>
<tr>
<td>$p_3 = 0.2$</td>
<td>$w_3 = 6$</td>
</tr>
</tbody>
</table>

- Stop all tasks after $w_1$: $R_1 = \frac{p_1}{w_1} = \frac{1}{30}$
Scheduling strategies

<table>
<thead>
<tr>
<th>Probability</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1 = 0.1$</td>
<td>$w_1 = 3$</td>
</tr>
<tr>
<td>$p_2 = 0.7$</td>
<td>$w_2 = 5$</td>
</tr>
<tr>
<td>$p_3 = 0.2$</td>
<td>$w_3 = 6$</td>
</tr>
</tbody>
</table>

- Stop all tasks after $w_1$: $R_1 = \frac{p_1}{w_1} = \frac{1}{30}$
- Stop all tasks after $w_2$: $R_2 = \frac{p_1 + p_2}{p_1 w_1 + (1-p_1)w_2} = \frac{1}{6}$
### Scheduling strategies

<table>
<thead>
<tr>
<th>Probability</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1 = 0.1$</td>
<td>$w_1 = 3$</td>
</tr>
<tr>
<td>$p_2 = 0.7$</td>
<td>$w_2 = 5$</td>
</tr>
<tr>
<td>$p_3 = 0.2$</td>
<td>$w_3 = 6$</td>
</tr>
</tbody>
</table>

- Stop all tasks after $w_1$: $R_1 = \frac{p_1}{w_1} = \frac{1}{30}$
- Stop all tasks after $w_2$: $R_2 = \frac{p_1 + p_2}{p_1 w_1 + (1-p_1)w_2} = \frac{1}{6}$
- Stop half unsuccessful tasks after $w_1$ and one-third after $w_2$: $R =$?
Optimal strategy

Theorem
Best strategy is to stop all tasks at some threshold

Strategy
Find $i$ maximizing

$$\mathcal{R}_i \overset{\text{def}}{=} \frac{\sum_{j=1}^{i} p_j}{\sum_{j=1}^{i} p_j w_j + (1 - \sum_{j=1}^{i} p_j) w_i}$$

If ties, pick smallest index
### Scheduling strategies

- Stop all tasks after $w_1$: $R_1 = \frac{p_1}{w_1} = \frac{1}{30}$
- Stop all tasks after $w_2$: $R_2 = \frac{p_1 + p_2}{p_1 w_1 + (1-p_1)w_2} = \frac{1}{6}$
- Stop all tasks after $w_3$: $R_3 = \frac{1}{p_1 w_1 + p_2 w_2 + p_3 w_3} = \frac{1}{5}$

<table>
<thead>
<tr>
<th>Probability</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1 = 0.1$</td>
<td>$w_1 = 3$</td>
</tr>
<tr>
<td>$p_2 = 0.7$</td>
<td>$w_2 = 5$</td>
</tr>
<tr>
<td>$p_3 = 0.2$</td>
<td>$w_3 = 6$</td>
</tr>
</tbody>
</table>
Scheduling strategies

<table>
<thead>
<tr>
<th>Probability</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1 = 0.1$</td>
<td>$w_1 = 3$</td>
</tr>
<tr>
<td>$p_2 = 0.7$</td>
<td>$w_2 = 5$</td>
</tr>
<tr>
<td>$p_3 = 0.2$</td>
<td>$w_3 = 6$</td>
</tr>
</tbody>
</table>

**Question?**

So in the end you should not interrupt anything, right? Pfhhhh these scheduling guys 😞

- Stop all tasks after $w_2$: $R_2 = \frac{p_1 + p_2}{p_1 w_1 + (1-p_1)w_2} = \frac{1}{6}$
- Stop all tasks after $w_3$: $R_3 = \frac{1}{p_1 w_1 + p_2 w_2 + p_3 w_3} = \frac{1}{5}$
Scheduling strategies

<table>
<thead>
<tr>
<th>Probability</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1 = 0.1$</td>
<td>$w_1 = 3$</td>
</tr>
<tr>
<td>$p_2 = 0.7$</td>
<td>$w_2 = 5$</td>
</tr>
<tr>
<td>$p_3 = 0.2$</td>
<td>$w_3 = 101$</td>
</tr>
</tbody>
</table>

- Stop all tasks after $w_1$: $R_1 = \frac{p_1}{w_1} = \frac{1}{30}$
- Stop all tasks after $w_2$: $R_2 = \frac{p_1 + p_2}{p_1 w_1 + (1-p_1)w_2} = \frac{1}{6}$
- Stop all tasks after $w_3$: $R_3 = \frac{1}{p_1 w_1 + p_2 w_2 + p_3 w_3} = \frac{1}{24}$
From discrete to continuous distributions

- \( f(x) \) probability density, \( F(x) \) cumulative distribution
- Expected value \( \mu_D \), variance, \( \sigma_D^2 \)

\[
\begin{align*}
\text{arg max } R_i & \overset{\text{def}}{=} \frac{\sum_{j=1}^{i} p_j}{\sum_{j=1}^{i} p_j w_j + (1 - \sum_{j=1}^{i} p_j) w_i} \\
\text{arg max } R(l) & \overset{\text{def}}{=} \frac{F(l)}{\int_0^l x f(x) dx + (1 - F(l)) l}
\end{align*}
\]

No more a theorem, but hopefully a good heuristic . . .
Best cutting threshold

\[ D = \text{EXP}(\lambda) \]
Best cutting threshold

- $\mathcal{D} = \text{EXP}(\lambda)$
  - Interrupt at any instant ($\mathcal{R}_i$ constant)

- $\mathcal{D} = \text{UNIFORM}[a, b]$
Best cutting threshold

- $\mathcal{D} = \text{EXP}(\lambda)$
  Interrupt at any instant ($\mathcal{R}_l$ constant)

- $\mathcal{D} = \text{UNIFORM}[a, b]$
  Never interrupt ($\mathcal{R}_l$ maximal for $l = b$)
Best cutting threshold

Question?

Well, do you know any important distribution for which it is really worth interrupting tasks?

Pfhhh these scheduling guys 😞

- \( D = \text{UNIFORM}[a, b] \)
  - Never interrupt (\( R_t \) maximal for \( l = b \))
### A few distributions . . .

<table>
<thead>
<tr>
<th>Name</th>
<th>PDF</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>$\frac{1}{b-a}$</td>
<td></td>
</tr>
<tr>
<td>Exponential</td>
<td>$\lambda e^{-\lambda x}$</td>
<td></td>
</tr>
<tr>
<td>Half-normal</td>
<td>$\frac{\sqrt{2}}{\theta \sqrt{\pi}} e^{-\frac{x^2}{2\theta^2}}$</td>
<td></td>
</tr>
<tr>
<td>Lognormal</td>
<td>$\frac{1}{x\beta \sqrt{2\pi}} e^{-\frac{(\log(x)-\alpha)^2}{2\beta^2}}$</td>
<td></td>
</tr>
<tr>
<td>Beta</td>
<td>$\frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$</td>
<td></td>
</tr>
<tr>
<td>Gamma</td>
<td>$\frac{1}{\Gamma(k)\theta^k} x^{k-1} e^{-\frac{x}{\theta}}$</td>
<td></td>
</tr>
<tr>
<td>Weibull</td>
<td>$\frac{k}{\theta^k} x^{k-1} e^{-\left(\frac{x}{\theta}\right)^k}$</td>
<td></td>
</tr>
<tr>
<td>Inverse-gamma</td>
<td>$\frac{\theta^k}{\Gamma(k)} x^{-k-1} e^{-\frac{\theta}{x}}$</td>
<td></td>
</tr>
</tbody>
</table>
...and their optimal cutting threshold

![Graphs showing efficiency vs cutting threshold for different distributions]

- Beta(2, 2)
- Gamma(2, 0.5)
- Weibull(2, 1/Γ(1.5))
- Inv-Gamma(3, 2)
- Beta(0.5, 0.5)
- Gamma(0.5, 2)
- Weibull(0.5, 1/Γ(3))
- Inv-Gamma(1.5, 0.5)
- U(0, 1)
- Exp(1)
- |N(0, 1)|
- Lognormal(0, 1)
Heuristics with 1 processor

- **MeanVariance**$(x)$: kill a task as time $\mu_D + x\sigma_D$, with $x$ some constant
- **Quantile**$(x)$: kill a task when execution time reaches the $x$-quantile of $D$, with $0 \leq x \leq 1$
- **OptRatio**: optimal cutting threshold
Heuristics with many processors

- With budget $b$ and deadline $d$, enroll $\left\lceil \frac{b}{d} \right\rceil$ processors
- Run previous heuristics in parallel
Normalized for $\mu = 1$, budget and deadline $b = d = 100$

Exponential: $\lambda = 1$, $l_{\text{opt}} = 2$ (arbitrarily)

Uniform: $a = 0$, $b = 2$, $l_{\text{opt}} = 2$

Lognormal: $\alpha \approx -1.15$, $\beta \approx 1.52$, $\mu = 1$, $\sigma = 3$, $l_{\text{opt}} \approx 0.1$
Focus on **LogNormal**

Lognormal: $\alpha \approx -1.15$, $\beta \approx 1.52$, $\mu = 1$, $\sigma = 3$, $l_{\text{opt}} \approx 0.1$

First row $b = d = 100$, second row $b = d$, third row $b = 100$

(hence $\left\lceil \frac{b}{d} \right\rceil$ processors)
Cut them short

\[ \mu = 0.5 \text{ for Beta, } \mu = 1 \text{ for Gamma} \]

Cutting threshold is 0.01 for OR in both plots.

\[ b = d = 100 \]
Zoom on threshold impact

**Heuristics**

- OR(0.001)
- OR(0.002)
- OR(0.005)
- OR(0.01)
- OR(0.02)
- OR(0.05)
- OR(0.1)

**Beta(0.5,0.5)**

<table>
<thead>
<tr>
<th>Heuristics</th>
<th>Successful tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>OR(0.001)</td>
<td></td>
</tr>
<tr>
<td>OR(0.002)</td>
<td></td>
</tr>
<tr>
<td>OR(0.005)</td>
<td></td>
</tr>
<tr>
<td>OR(0.01)</td>
<td></td>
</tr>
<tr>
<td>OR(0.02)</td>
<td></td>
</tr>
<tr>
<td>OR(0.05)</td>
<td></td>
</tr>
<tr>
<td>OR(0.1)</td>
<td></td>
</tr>
</tbody>
</table>

**Gamma(0.5,2)**

<table>
<thead>
<tr>
<th>Heuristics</th>
<th>Successful tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>OR(0.001)</td>
<td></td>
</tr>
<tr>
<td>OR(0.002)</td>
<td></td>
</tr>
<tr>
<td>OR(0.005)</td>
<td></td>
</tr>
<tr>
<td>OR(0.01)</td>
<td></td>
</tr>
<tr>
<td>OR(0.02)</td>
<td></td>
</tr>
<tr>
<td>OR(0.05)</td>
<td></td>
</tr>
<tr>
<td>OR(0.1)</td>
<td></td>
</tr>
</tbody>
</table>

\[
\mu = 0.5 \text{ for } \text{Beta}, \quad \mu = 1 \text{ for } \text{Gamma}
\]

\[
b = d = 100
\]
Parallelism matters

<table>
<thead>
<tr>
<th>Probability</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1 = 0.4$</td>
<td>$w_1 = 2$</td>
</tr>
<tr>
<td>$p_2 = 0.15$</td>
<td>$w_2 = 3$</td>
</tr>
<tr>
<td>$p_3 = 0.45$</td>
<td>$w_3 = 7$</td>
</tr>
</tbody>
</table>

Budget $b = 6$, no deadline (say $d = 6$)
Optimal schedule with 1 processor

<table>
<thead>
<tr>
<th>Probability</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1 = 0.40$</td>
<td>$w_1 = 2$</td>
</tr>
<tr>
<td>$p_2 = 0.15$</td>
<td>$w_2 = 3$</td>
</tr>
</tbody>
</table>

- $E(1) = 0, E(2) = p_1 = 0.4$
- $E(3) = (p_1 + p_2) = 0.55$ (pointless to kill an unsuccessful task at time 2)
- $E(4) = \max\{p_1 + E(2), p_1(1 + E(2)) + p_2(1 + E(1))\} = 0.8$ Either kill the first task (if not completed) at time 2 or continue up to time 3 (if not completed) and then kill
- $E(6) = \max\{p_1 + E(4), p_1(1 + E(4)) + p_2(1 + E(3))\} = 1.2$
An efficient schedule with 2 processors

Two processors, each starting a task in parallel
If none completes by time 2, let them run up to time 3
Otherwise, kill at time 2 any not-yet completed task and start a new task

<table>
<thead>
<tr>
<th>Probability</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1 = 0.40$</td>
<td>$w_1 = 2$</td>
</tr>
<tr>
<td>$p_2 = 0.15$</td>
<td>$w_2 = 3$</td>
</tr>
</tbody>
</table>

Processor 1

<table>
<thead>
<tr>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$w_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_1$</td>
<td>$2 + p_1$</td>
<td>$1 + p_1$</td>
</tr>
<tr>
<td>$w_2$</td>
<td>$1 + p_1$</td>
<td>$2$</td>
</tr>
<tr>
<td>$w_3$</td>
<td>$1 + p_1$</td>
<td>$1$</td>
</tr>
</tbody>
</table>

Processor 2

With probability $p_1 p_2$, 1st task completes, 2nd task is killed, 2 units remain for the new one, expected number of completed tasks in this configuration is $1 + p_1$

\[
E_{//} = p_1^2 (2 + p_1) + 2p_1(p_2 + p_3)(1 + p_1) + 2p_2^2 + 2p_2 p_3 = 1.236
\]
Single processor, no deadline ($d = b$), without preemption

**Function** SeqSched($\beta, s$)
- **Data:** The budget $\beta$
- The threshold $s$ at which the last executed task stopped ($s = 0$ if the execution was successful)
- $bestExpectation \leftarrow 0$
- /* If the budget allows it, we can attempt to start a new task */
- if $\beta \geq w_1$ then
  - $bestExpectation \leftarrow p_1(1 + SeqSched(\beta - w_1, 0)) + (1 - p_1)(SeqSched(\beta - w_1, 1))$
- /* If there was a task preempted at threshold $s$ and

**Without preemption**

**Complexity $O(kb)$**

**Pseudo-polynomial 😞**

\[
\frac{1 - \frac{1}{\sum p_i}}{1 - \sum p_i} (SeqSched(\beta - (w_{s+1} - w_s), s + 1))
\]

$bestExpectation \leftarrow \max\{bestExpectation, expectation\}$

return $bestExpectation$

return $SeqSched(b, 0)$
The little scheduling problem

Core instance

Experiments

Parallelism matters

Complexity results

Single processor, no deadline \((d = b)\), with preemption

Function `PSeqSched(\beta, S)`

Data: The budget \(\beta\)

An array \(S\) of size \(k\): \(S[i]\) is the number of tasks preempted at state \(i\)

\begin{align*}
  & \text{bestExpectation} \leftarrow 0 \\
  & /* If the budget allows it, we can attempt to start a new task */ \text{if } \beta \geq w_1 \text{ then} \\
  & \quad \text{bestExpectation} \leftarrow p_1(1 + PSeqSched(\beta - w_1, S)) + (1 - p_1)(PSeqSched(\beta - w_1, S + 1)) \\
  & \text{for } s = 1 \text{ to } k - 1 \text{ do} \\
  & \quad /* If there was a task preempted at threshold } s \text{ and} \\
  & \text{let } S \text{ be an array of size } k - 1 \text{ with } S[i] = 0 \text{ for all } i \\
  & \text{return } PSeqSched(b, S)
\end{align*}

With preemption

Complexity \(O\left(\prod_{s=1}^{k-1} \left(1 + \frac{b}{w_s}\right)\right)\)

Pseudo-polynomial only for fixed \(k\) 😞
Parallel processors, no deadline, without preemption

```
Function ParSchedDecision(β, T1, T2)
    Data: The budget β
    A set T1: T1[i] is the progress of a task that may be interrupted
    A set T2: T2[i] is the progress of a task that cannot be
    interrupted
    if β = 0 then return 0
    if T1 = ∅ then
        q ← ⌊β/w1⌋
        /* In addition to the current progressing tasks,
        we can start new ones */
        return max0≤q≤q ParSchedJump(β, T2 ∪ {0})
    else
        /* Task 1 in T1 is either interrupted or not */
        return max(ParSchedDecision(β, T1 \ {T1[1]}, T2),
                   ParSchedDecision(β, T1 \ {T1[1]}, T2 ∪ {T1[1]}))

Function ParSchedJump(β, T)
    Data: The budget β
    A set T: T[i] is the progress of a task
    if T = ∅ then return 0
```

Without preemption

\[ q \leftarrow \lfloor \frac{\beta}{w_1} \rfloor \]

Time complexity \( b(q + k)q^2 w_k^q = O((b + k)b^3 w_k^b) \)

Space complexity \( bq^2 w_k^q = O(b^3 w_k^b) \)

Very expensive 😞
Relations between problems

- Any algorithm on $p$ processors with/without preemption can be simulated on a single processor with preemption with same performance
- ParSched is never worse than SeqSched, and can achieve strictly better performance on some problem instances
- PSeqSched is never worse than ParSched, and can achieve strictly better performance on some problem instances
Conclusion

- Many complexity results
- Asymptotic algorithm for large budgets and deadlines

- Future work: heterogeneous processors

- Collaborations?
  Models for job execution time?