Universal Scheduling on a single machine

Leah Epstein, Asaf Levin, Alberto Marchetti Spaccamela, Nicole Megow,
Julian Mestre, Martin Skutella, Leen Stougie

Scheduling

Given a set of jobs and a set of machines.

- Each job has a processing time.
- Each job has to be processed by one of the machines
- Each machine can process at most one job at a time
- A schedule is an assignment of the jobs to the machine together with an order of the jobs on the machines
- Given a schedule the time at which a job completes is called its completion time
- Objectives are functions of the completion times of jobs like maximum completion time or average completion times
- There may be deadlines, release times, precedence constraints etc.

Scheduling

Given a set J of n jobs to be scheduled on 1 machine

- job j has processing time p_j and weight w_j
- Objective: Minimize the weighted sum of completion times
  $1 \mid\mid \sum w_j C_j$

Robust Scheduling

General scheduling policies assume machines are always available or unavailability periods are known

Robust Scheduling

General scheduling policies assume machines are always available or unavailability periods are known

What about scheduling policies able to cope with unexpected machines breakdown or slowdown?

The scheduler has usually no control over disruptions and does not know beforehand when they occur

Robust Scheduling

General scheduling policies assume machines are always available or unavailability periods are known

What about scheduling policies able to cope with unexpected machines breakdown or slowdown?

The scheduler has usually no control over disruptions and does not know beforehand when they occur

We are interested in scheduling policies that are robust with respect to machine break- and slow-down

Measure of Robustness

The quality of the robust algorithm is measured by the worst-case ratio with the schedule of an optimal algorithm having full knowledge of the machines availability

Measure of Robustness

The quality of the robust algorithm is measured by the worst-case ratio with the schedule of an optimal algorithm having full knowledge of the machines availability

The algorithm decides on the scheduling policy before the execution starts

Measure of Robustness

The quality of the robust algorithm is measured by the worst-case ratio with the schedule of an optimal algorithm having full knowledge of the machines availability

The algorithm decides on the scheduling policy before the execution starts

During execution the algorithm does not change the scheduling policy
Different from on-line algorithm

Measure of Robustness

The quality of the robust algorithm is measured by the worst-case ratio with the schedule of an optimal algorithm having full knowledge of the machines availability

The algorithm decides on the scheduling policy before the execution starts

During execution the algorithm does not change the scheduling policy
Different from on-line algorithm

A robust algorithm fixes the sequence of the jobs

Measure of Robustness

The quality of the robust algorithm is measured by the worst-case ratio with the schedule of an optimal algorithm having full knowledge of the machines availability

The algorithm decides on the scheduling policy before the execution starts

During execution the algorithm does not change the scheduling policy
Different from on-line algorithm

A robust algorithm fixes the sequence of the jobs

Also called in the literature universal scheduling

Alternative uncertainty models

Extremes: Off-line $\leftrightarrow$ Robust

Uncertainty: which of a given set of scenarios may occur

Alternative uncertainty models

Extremes: Off-line $\leftrightarrow$ Robust

Uncertainty: which of a given set of scenarios may occur

	Adaptive	Nonadaptive
Deterministic	On-line/Off-line	Robust scheduling
Stochastic	2-stage optimization with just 2nd stage (probabilistic analysis)	2-stage optimization decision made in stage 1 (a priori scheduling)

Alternative uncertainty models

Extremes: Off-line $\leftrightarrow$ Robust

Uncertainty: which of a given set of scenarios may occur

	Adaptive	Nonadaptive
Deterministic	On-line/Off-line	Robust scheduling
Stochastic	2-stage optimization with just 2nd stage (probabilistic analysis)	2-stage optimization decision made in stage 1 (a priori scheduling)

The problem $1 \mid\mid \sum w_j C_j$

Given a set J of n jobs to be scheduled on 1 machine

- job j has processing time p_j and weight w_j
- Objective: Minimize the weighted sum of completion times $1 \mid\mid \sum w_j C_j$

The problem $1 \mid\mid \sum w_j C_j$

Given a set J of n jobs to be scheduled on 1 machine

- job j has processing time p_j and weight w_j
- Objective: Minimize the weighted sum of completion times $1 \mid\mid \sum w_j C_j$

- Machine slows down or becomes (temporary) unavailable
- Preemption is allowed

The problem $1 \mid\mid \sum w_j C_j$

Given a set J of n jobs to be scheduled on 1 machine

- job j has processing time p_j and weight w_j
- Objective: Minimize the weighted sum of completion times
  $1 \mid\mid \sum w_j C_j$
- Machine slows down or becomes (temporary) unavailable
- Preemption is allowed
  Specifically, the job being processed at the moment of a machine break down can be resumed without loss of processing as soon as the machine becomes available again

The problem $1 \mid\mid \sum w_j C_j$

Complexity of the non-robust version

- No disruption: simple polynomial time optimal algorithm: schedule jobs according to increasing ratio p_j/w_j

The problem $1 || \sum w_j C_j$

Complexity of the non-robust version

- No disruption: simple polynomial time optimal algorithm: schedule jobs according to increasing ratio p_j / w_j
- weakly NP-complete with 1 unavailable interval (KNAPSACK)

The problem $1 \mid\mid \sum w_j C_j$

Complexity of the non-robust version

- No disruption: simple polynomial time optimal algorithm: schedule jobs according to increasing ratio p_j/w_j
- weakly NP-complete with 1 unavailable interval
- strongly NP-complete 2 or more unavailable periods

The problem $1 \mid\mid \sum w_j C_j$

Complexity of the non-robust version

- No disruption: simple polynomial time optimal algorithm: schedule jobs according to increasing ratio p_j/w_j
- weakly NP-complete with 1 unavailable interval
- strongly NP-complete 2 or more unavailable periods
- in P if all $w_j = 1$

The problem $1 \mid\mid \sum w_j C_j$

Complexity of the non-robust version

- No disruption: simple polynomial time optimal algorithm: schedule jobs according to increasing ratio p_j/w_j
- weakly NP-complete with 1 unavailable interval
- strongly NP-complete 2 or more unavailable periods
- in P if all $w_j = 1$
- Without preemption two or more unavailbility periods is hard even if all $w_j = 1$

No robust schedule is always optimal

2 jobs: job 1 has $p_1 = 2$, $w_1 = 3$; job 2 has $p_2 = 1$, $w_2 = 1$

No robust schedule is always optimal

2 jobs: job 1 has $p_1 = 2$, $w_1 = 3$; job 2 has $p_2 = 1$, $w_2 = 1$

- No disruption: optimum schedule is first 1 then 2
  - Schedule 1,2 has cost $6 + 3 = 9$
  - Schedule 2,1 has cost $1 + 9 = 10$

No robust schedule is always optimal

2 jobs: job 1 has $p_1 = 2$, $w_1 = 3$; job 2 has $p_2 = 1$, $w_2 = 1$

- No disruption: optimum schedule is first 1 then 2
  - Schedule 1,2 has cost $6 + 3 = 9$
  - Schedule 2,1 has cost $1 + 9 = 10$
- If machine is not available between time 1 and time 10 then
  - Schedule 1,2 has cost $3 \cdot 11 + 1 \cdot 12 = 45$
  - Schedule 2,1 has cost $1 \cdot 1 + 3 \cdot 12 = 37$

No robust schedule is always optimal

2 jobs: job 1 has $p_1 = 2$, $w_1 = 3$; job 2 has $p_2 = 1$, $w_2 = 1$

- No disruption: optimum schedule is first 1 then 2
  - Schedule 1,2 has cost $6 + 3 = 9$
  - Schedule 2,1 has cost $1 + 9 = 10$
- If machine is not available between time 1 and time 10 then
  - Schedule 1,2 has cost $3 \cdot 11 + 1 \cdot 12 = 45$
  - Schedule 2,1 has cost $1 \cdot 1 + 3 \cdot 12 = 37$
- If machine is not available between time 1 and 2 then optimal schedule is again first 1 and then 2

Price of Robustness

PRICE OF ROBUSTNESS: The worst-case ratio that any robust algorithm has to incur with respect to a clairvoyant optimum

No robust schedule is always optimal

2 jobs: job 1 has $p_1 = 2$, $w_1 = 3$; job 2 has $p_2 = 1$, $w_2 = 1$

- No disruption: optimum schedule is first 1 then 2
  - Schedule 1,2 has cost 6+ 3= 9
  - Schedule 2,1 has cost 1 + 9=10
- If machine is not available between time 1 and time 10 then
  - Schedule 1,2 has cost 33+12=45
  - Schedule 2,1 has cost 1+ 36= 37
- If machine is not available between time 1 and 2 then optimal schedule is again first 1 and then 2

Price of Robustness is $10/9$

Machine capacity function

The machine capacity function $f(t)$ is the aggregated amount of processing time available up to time t

Robustness against any machine capacity function

Results

- A deterministic polynomial time algorithm that provides a sequence that for any $\epsilon > 0$ and any machine capacity function is at most $4 + \epsilon$ times the optimum
- e-approximate randomized algorithm
- lower bound of 4 for deterministic algorithms
- lower bound of e for randomized algorithms
- Extension to (restricted) precedence constraints
- FPTAS for 1 unavailability problem (non-robust)

Deterministic $4+\epsilon$

Assume maximum speed 1. Let π be job-order and $S(\pi)$ a resulting schedule

- $C_j^{S(\pi)} = \min\{t \mid f(t) \geq C_j^\pi\}$ completion time of job j
- $W^{S(\pi)}(t)$ total weight of jobs not completed by time t

Deterministic $4+\epsilon$

Assume maximum speed 1. Let π be job-order and $S(\pi)$ a resulting schedule

- $C_j^{S(\pi)} = \min\{t \mid f(t) \geq C_j^\pi\}$ completion time of job j
- $W^{S(\pi)}(t)$ total weight of jobs not completed by time t
- If there are no breaks (speed-1 for all t) then

$$\sum_{j \in J} w_j C_j^{S(\pi)} = \sum_{j \in J} w_j C_j^\pi = \int_0^\infty W^\pi(t)\mathrm{d}t$$

Deterministic $4+\epsilon$

Assume maximum speed 1. Let π be job-order and $S(\pi)$ a resulting schedule

- $C_j^{S(\pi)} = \min\{t \mid f(t) \geq C_j^\pi\}$ completion time of job j
- $W^{S(\pi)}(t)$ total weight of jobs not completed by time t
- If there are no breaks (speed-1 for all t) then

$$\sum_{j\in J} w_j C_j^{S(\pi)} = \sum_{j\in J} w_j C_j^\pi = \int_0^\infty W^\pi(t)\mathrm{d}t$$

- With $f(t)$ the machine capacity function

$$\sum_{j\in J} w_j C_j^{S(\pi)} = \int_0^\infty W^{S(\pi)}(t)\mathrm{d}t = \int_0^\infty W^\pi(f(t))\mathrm{d}t$$

Deterministic $4 + \epsilon$

- Given a time instant t let $W^*(t) = \min_\pi \; W^\pi(t)$

Deterministic $4 + \epsilon$

- Given a time instant t let $W^*(t) = \min_\pi \; W^\pi(t)$
- For any f, a lower bound on the objective value of any schedule is

$$\int_0^\infty W^*(f(t))$$

Deterministic $4 + \epsilon$

- Given a time instant t let $W^*(t) = \min_\pi \ W^\pi(t)$
- For any f, a lower bound on the objective value of any schedule is

$$\int_0^\infty W^*(f(t))$$

- KEY LEMMA. *A sequence π of jobs yields a schedule with objective value at most c times the optimum value for all machine capacity functions f if and only if for all t*

$$W^\pi(t) \leq cW^*(t)$$

Deterministic $4 + \epsilon$

KEY LEMMA. *A sequence* π *of jobs yields a schedule with objective value at most* c *times the optimum value for all machine capacity functions* f *if and only if for all* t

$$W^{\pi}(t) \leq cW^{*}(t)$$

Deterministic $4 + \epsilon$

KEY LEMMA. *A sequence* π *of jobs yields a schedule with objective value at most* c *times the optimum value for all machine capacity functions* f *if and only if for all* t

$$W^{\pi}(t) \leq cW^{*}(t)$$

"if" is clear, since

$$\sum_{j \in J} w_j C_j^{S(\pi)} = \int_0^{\infty} W^{\pi}(f(t))\mathrm{d}t \leq c \int_0^{\infty} W^{*}(f(t))\mathrm{d}t$$

Deterministic $4 + \epsilon$

KEY LEMMA. *A sequence* π *of jobs yields a schedule with objective value at most* c *times the optimum value for all machine capacity functions* f *if and only if for all* t

$$W^\pi(t) \leq cW^*(t)$$

“only if” by contradiction. Suppose $W^\pi(t_0) > cW^*(t_0)$ for some t_0.

Deterministic $4 + \epsilon$

KEY LEMMA. *A sequence* π *of jobs yields a schedule with objective value at most* c *times the optimum value for all machine capacity functions* f *if and only if for all* t

$$W^\pi(t) \leq cW^*(t)$$

"only if" by contradiction. Suppose $W^\pi(t_0) > cW^*(t_0)$ for some t_0.

$$f(t) = \begin{cases} t & \text{if } t \leq t_0 \\ t_0 & \text{if } t_0 < t \leq t_1 \\ t - (t_1 - t_0) & \text{if } t > t_1 \end{cases}$$

breakdown in $[t_0, t_1]$

Deterministic $4 + \epsilon$

$$f(t) = \begin{cases} t & \text{if } t \leq t_0 \\ t_0 & \text{if } t_0 < t \leq t_1 \\ t - (t_1 - t_0) & \text{if } t > t_1 \end{cases}$$

breakdown in $[t_0, t_1]$

$$\sum_{j \in J} w_j C_j^{S(\pi)} = \sum_{j \in J} w_j C_j^{\pi} + (t_1 - t_0) W^{\pi}(t_0)$$

Deterministic $4 + \epsilon$

$$f(t) = \begin{cases} t & \text{if } t \leq t_0 \\ t_0 & \text{if } t_0 < t \leq t_1 \\ t - (t_1 - t_0) & \text{if } t > t_1 \end{cases}$$

breakdown in $[t_0, t_1]$

$$\sum_{j \in J} w_j C_j^{S(\pi)} = \sum_{j \in J} w_j C_j^{\pi} + (t_1 - t_0) W^{\pi}(t_0)$$

For π^* with $W^{\pi^*}(t_0) = W^*(t_0)$

$$\sum_{j \in J} w_j C_j^{S(\pi^*)} = \sum_{j \in J} w_j C_j^{\pi^*} + (t_1 - t_0) W^*(t_0)$$

Deterministic $4+\epsilon$

$$f(t)=\begin{cases} t & \text{if } t\le t_0\\ t_0 & \text{if } t_0<t\le t_1\\ t-(t_1-t_0) & \text{if } t>t_1\end{cases}$$

breakdown in $[t_0,t_1]$

$$\sum_{j\in J} w_j C_j^{S(\pi)} = \sum_{j\in J} w_j C_j^{\pi} + (t_1-t_0)W^{\pi}(t_0)$$

For π^* with $W^{\pi^*}(t_0)=W^*(t_0)$

$$\sum_{j\in J} w_j C_j^{S(\pi^*)} = \sum_{j\in J} w_j C_j^{\pi^*} + (t_1-t_0)W^{*}(t_0)$$

Since by assumption $W^{\pi}(t_0)/W^*(t_0)>c$ making $t_1\to\infty$ causes $\sum_{j\in J} w_j C_j^{S(\pi)}/\sum_{j\in J} w_j C_j^{S(\pi^*)}>c$. Contradiction

Deterministic $4 + \epsilon$

Given our Key Lemma, we can use a result of Bechetti, Leonardi, Marchetti Spaccamela and Pruhs, 2003 who design an algorithm for which they prove $W^{\pi}(t) \leq 24W^{*}(t)$ for all t showing that a constant ratio (price of robustness) exists.

We will design an algorithm with $c = 4$

Deterministic $4 + \epsilon$

Idea: put jobs with small w and large p later

Algorithm A.

- Given a set J of jobs construct the sequence backwards
- Iteration i, $i = 0, 1, 2, \ldots$: Let J_i be the set of jobs with total weight $\leq 2^i$ and maximum total processing time
- Schedule jobs in

$$J_i \setminus \bigcup_{j<i} J_j$$

in any order at the end, just before all jobs in J_{i-1}

Deterministic $4 + \epsilon$

Idea: put jobs with small w and large p later

Algorithm A.

- Given a set J of jobs construct the sequence backwards
- Iteration i, $i = 0, 1, 2, \ldots$: Let J_i be the set of jobs with total weight $\leq 2^i$ and maximum total processing time
- Schedule jobs in

$$J_i \setminus \bigcup_{j<i} J_j$$

in any order at the end, just before all jobs in J_{i-1}

Note: Algorithm A is not polynomial time

Theorem: A is robust 4-approximate

Proof: by the Key Lemma it is sufficient to show that $W^{\pi}(t) \leq 4W^{*}(t)$ for all t.

- Let $p(J_i)$ total processing time of jobs in J_i
- Let $w(J_i)$ total weight of jobs in J_i

Theorem: A is robust 4-approximate

Proof: by the Key Lemma it is sufficient to show that $W^\pi(t) \leq 4W^*(t)$ for all t.

- Let $p(J_i)$ total processing time of jobs in J_i
- Let $w(J_i)$ total weight of jobs in J_i

Given t let k minimum integer s.t. $p(J_k) \geq p(J) - t$

Theorem: A is robust 4-approximate

Proof: by the Key Lemma it is sufficient to show that $W^\pi(t) \leq 4W^*(t)$ for all t.

- Let $p(J_i)$ total processing time of jobs in J_i
- Let $w(J_i)$ total weight of jobs in J_i

Given t let k minimum integer s.t. $p(J_k) \geq p(J) - t$

By construction of π, $C_j^\pi > t$ only if $j \in \cup_{i=0}^{k} J_i$.

Theorem: A is robust 4-approximate

Proof: by the Key Lemma it is sufficient to show that $W^\pi(t) \leq 4W^*(t)$ for all t.

- Let $p(J_i)$ total processing time of jobs in J_i
- Let $w(J_i)$ total weight of jobs in J_i

Given t let k minimum integer s.t. $p(J_k) \geq p(J) - t$

By construction of π, $C_j^\pi > t$ only if $j \in \cup_{i=0}^k J_i$. Thus,

$$(1) \qquad W^\pi(t) \leq \sum_{i=0}^{k} w(J_i) \leq \sum_{i=0}^{k} 2^i = 2^{k+1} - 1 \tag{1}$$

Theorem: A is robust 4-approximate

Proof: by the Key Lemma it is sufficient to show that $W^\pi(t) \leq 4W^*(t)$ for all t.

Given t let k minimum integer s.t. $p(J_k) \geq p(J) - t$

$$W^\pi(t) \leq \sum_{i=0}^{k} w(J_i) \leq \sum_{i=0}^{k} 2^i = 2^{k+1} - 1 \qquad (1)$$

Theorem: A is robust 4-approximate

Proof: by the Key Lemma it is sufficient to show that $W^\pi(t) \le 4W^*(t)$ for all t.

Given t let k minimum integer s.t. $p(J_k) \ge p(J) - t$

$$W^\pi(t) \le \sum_{i=0}^{k} w(J_i) \le \sum_{i=0}^{k} 2^i = 2^{k+1} - 1 \qquad (1)$$

If $k = 0$ the statement is clearly true

Theorem: A is robust 4-approximate

Proof: by the Key Lemma it is sufficient to show that $W^{\pi}(t) \leq 4W^{*}(t)$ for all t.

Given t let k minimum integer s.t. $p(J_k) \geq p(J) - t$

$$W^{\pi}(t) \leq \sum_{i=0}^{k} w(J_i) \leq \sum_{i=0}^{k} 2^i = 2^{k+1} - 1 \qquad (1)$$

Choice of k implies $p(J_{k-1}) < p(J) - t$.

Theorem: A is robust 4-approximate

Proof: by the Key Lemma it is sufficient to show that $W^{\pi}(t) \leq 4W^{*}(t)$ for all t.

Given t let k minimum integer s.t. $p(J_k) \geq p(J) - t$

$$(1) \qquad W^{\pi}(t) \leq \sum_{i=0}^{k} w(J_i) \leq \sum_{i=0}^{k} 2^i = 2^{k+1} - 1$$

Choice of k implies $p(J_{k-1}) < p(J) - t$. Moreover, by maximality of $p(J_{k-1})$ any π' must have $W^{\pi'}(t) > 2^{k-1}$.

Theorem: A is robust 4-approximate

Proof: by the Key Lemma it is sufficient to show that $W^{\pi}(t) \leq 4W^{*}(t)$ for all t.

Given t let k minimum integer s.t. $p(J_k) \geq p(J) - t$

$$(1) \qquad W^{\pi}(t) \leq \sum_{i=0}^{k} w(J_i) \leq \sum_{i=0}^{k} 2^i = 2^{k+1} - 1$$

Choice of k implies $p(J_{k-1}) < p(J) - t$. Moreover, by maximality of $p(J_{k-1})$ any π' must have $W^{\pi'}(t) > 2^{k-1}$. Thus,

$$(2) \qquad W^{*}(t) \geq 2^{k-1}$$

Theorem: A is robust 4-approximate

Proof: by the Key Lemma it is sufficient to show that $W^{\pi}(t) \leq 4W^*(t)$ for all t.

Given t let k minimum integer s.t. $p(J_k) \geq p(J) - t$

$$(1) \qquad W^{\pi}(t) \leq \sum_{i=0}^{k} w(J_i) \leq \sum_{i=0}^{k} 2^i = 2^{k+1} - 1$$

Choice of k implies $p(J_{k-1}) < p(J) - t$. Moreover, by maximality of $p(J_{k-1})$ any π' must have $W^{\pi'}(t) > 2^{k-1}$. Thus,

$$(2) \qquad W^*(t) \geq 2^{k-1}$$

(1) and (2) combine to the result

A polynomial time algorithm

- Algorithm A requires at each iteration to solve a knapsack problem
- Instead of using exact solution use a $(1+\epsilon)$ approximation
- Hence we obtain a $(4+\epsilon)$ approximation with running time polynomial in the number of jobs and $1/\epsilon$

A randomised algorithm

- Randomised Alg: similar to deterministic algorithm
- Weight of jobs selected at iteration i
  - Deterministic: weights sequence $2^0, 2^1, \ldots$
  - Random: choose a random value y in $[0, 1]$ and then the sequence of weights is $e^{i+y},\ i = 0, 1, \ldots$
- **Theorem.** For every instance the randomized algorithm produces a random permutation ρ s.t.

$$E[W^\rho(t)] \leq eW^*(t),\ \forall\ t$$

Lower bound

Consider the following sequence of jobs:

$$p_k = k^k,\ w_k = k,\ k = M, M+1, \ldots, M^M$$

For any $k \geq M$ we have $\sum_{j<k} p_j < p_k$. Let $T = \sum_k p_k$

Lower bound

Consider the following sequence of jobs:

$$p_k = k^k,\ w_k = k,\ k = M, M+1, \ldots, M^M$$

For any $k \geq M$ we have $\sum_{j<k} p_j < p_k$. Let $T = \sum_k p_k$

Lemma. If the price of robustness (PoR) of a job order is smaller than α then any job $k > M\alpha$ must be succeeded (not necessarily directly) by some job r with $k/\alpha \leq r \leq k-1$

Lower bound

Consider the following sequence of jobs:

$$p_k = k^k,\ w_k = k,\ k = M, M+1, \ldots, M^M$$

For any $k \geq M$ we have $\sum_{j<k} p_j < p_k$. Let $T = \sum_k p_k$

Lemma. If the price of robustness (PoR) of a job order is smaller than α then any job $k > M\alpha$ must be succeeded (not necessarily directly) by some job r with $k/\alpha \leq r \leq k-1$

Given a schedule π with PoR: in backward direction starting from the last job in the schedule construct a maximal sequence of jobs with monotonically increasing weight, say length L. Let their weights starting with the last one be $w^1, w^2, \ldots, w^L$. By Lemma for all k, $w^{k+1}/w^k \leq \alpha$ and (for $\alpha < M$) $L \geq M-1$.

Lower bound

- $p_k = k^k, \ w_k = k, \ k = M, M+1, \ldots, M^M$

  For any $k \geq M$ we have $\sum_{j<k} p_j < p_k$. Let $T = \sum_k p_k$

- $w^1, w^2, \ldots, w^L; \ w^{k+1}/w^k \leq \alpha; \ L \geq M - 1$

Lower bound

- $p_k = k^k, \; w_k = k, \; k = M, M+1, \dots, M^M$

  For any $k \geq M$ we have $\sum_{j<k} p_j < p_k$. Let $T = \sum_k p_k$
- $w^1, w^2, \dots, w^L; \; w^{k+1}/w^k \leq \alpha; \; L \geq M-1$

Using w^k as the number of the job: At time $t_k = T - p(w^k+1)$ in the schedule π a.o. jobs $w^1, \dots, w^{k+1}$ are not completed. OPT has not completed job w^k+1. PoR of π is at least

$$\alpha \geq \frac{\sum_{j=1}^{k+1} w^j}{w^k+1}, \; \forall k = 1, \dots, M-1$$

The recurrence implies that $\alpha \geq 4$

QED

Release dates

Giving the order as before, ignoring release dates, and at any time processing the highest priority job in that order that is available satisfies

$$W^{\pi}(t) \le 4W^{*}(t), \; \forall \, t$$

Release dates

Giving the order as before, ignoring release dates, and at any time processing the highest priority job in that order that is available satisfies $W^\pi(t) \leq 4W^*(t), \ \forall t$

However, Key Lemma does not hold:

$r_i = i, \ p_i = w_i = 1, \ i = 1, \ldots, n$, $r_{n+1} = n+1, p_{n+1} = 1, w_{n+1} = M$.
Clearly the order $\pi = 1, 2, \ldots, n, n+1$ satisfies the inequality.

Release dates

Giving the order as before, ignoring release dates, and at any time processing the highest priority job in that order that is available satisfies $W^{\pi}(t) \leq 4W^*(t), \ \forall t$

However, Key Lemma does not hold:

$r_i = i, \ p_i = w_i = 1, \ i = 1, \ldots, n$, $r_{n+1} = n+1, p_{n+1} = 1, w_{n+1} = M$.
Clearly the order $\pi = 1, 2, \ldots, n, n+1$ satisfies the inequality.

Breakdown at $[n, n+1]$. According to π, job n is processed at time $n+1$. Long breakdown starting at time $t \in [n+2, n+3)$. Job $n+1$ uncompleted. In OPT job n is uncompleted, yielding ratio of $M/1 = M$.

Release dates

Giving the order as before, ignoring release dates, and at any time processing the highest priority job in that order that is available satisfies $W^\pi(t) \leq 4W^*(t), \ \forall t$

However, Key Lemma does not hold:

$r_i = i, \ p_i = w_i = 1, \ i = 1, \ldots, n$, $r_{n+1} = n+1, p_{n+1} = 1, w_{n+1} = M$.
Clearly the order $\pi = 1, 2, \ldots, n, n+1$ satisfies the inequality.

Breakdown at $[n, n+1]$. According to π, job n is processed at time $n+1$. Long breakdown starting at time $t \in [n+2, n+3)$. Job $n+1$ uncompleted. In OPT job n is uncompleted, yielding ratio of $M/1 = M$.

Theorem. The price of robustness in the presence of release dates is $\Omega(\frac{\log n}{\log \log n})$

Release dates

Giving the order as before, ignoring release dates, and at any time processing the highest priority job in that order that is available satisfies $W^{\pi}(t) \leq 4W^{*}(t),\ \forall t$

However, Key Lemma does not hold:

$r_i = i,\ p_i = w_i = 1,\ i = 1, \ldots, n$, $r_{n+1} = n+1, p_{n+1} = 1, w_{n+1} = M$.
Clearly the order $\pi = 1, 2, \ldots, n, n+1$ satisfies the inequality.

Breakdown at $[n, n+1]$. According to π, job n is processed at time $n+1$. Long breakdown starting at time $t \in [n+2, n+3)$. Job $n+1$ uncompleted. In OPT job n is uncompleted, yielding ratio of $M/1 = M$.

Theorem. The price of robustness in the presence of release dates is $\Omega(\frac{\log n}{\log \log n})$

(in fact through an adversarial instance with all unit weights $w_i = 1$)

Release dates

Giving the order as before, ignoring release dates, and at any time processing the highest priority job in that order that is available satisfies $W^\pi(t) \leq 4W^*(t),\ \forall t$

However, Key Lemma does not hold:

$r_i = i,\ p_i = w_i = 1,\ i = 1, \ldots, n$, $r_{n+1} = n+1, p_{n+1} = 1, w_{n+1} = M$.
Clearly the order $\pi = 1, 2, \ldots, n, n+1$ satisfies the inequality.

Breakdown at $[n, n+1]$. According to π, job n is processed at time $n+1$. Long breakdown starting at time $t \in [n+2, n+3)$. Job $n+1$ uncompleted. In OPT job n is uncompleted, yielding ratio of $M/1 = M$.

Theorem. The price of robustness in the presence of release dates is $\Omega(\frac{\log n}{\log \log n})$

No upperbound so far

Release dates: Lower bound

Theorem. The PoR in the presence of release dates is $\Omega(\frac{\log n}{\log \log n})$

Release dates: Lower bound

Theorem. The PoR in the presence of release dates is $\Omega(\frac{\log n}{\log \log n})$

Instance: $w_j = 1,\, p_j = 2^j,\, r_{n-1-j} = \sum_{i>j} 2^i = \sum_{i>j} p_i,\, j = 0, 1, \ldots, n-1$

Release dates: Lower bound

Theorem. The PoR in the presence of release dates is $\Omega(\frac{\log n}{\log \log n})$

Instance: $w_j = 1$, $p_j = 2^j$, $r_{n-1-j} = \sum_{i>j} 2^i = \sum_{i>j} p_i$, $j = 0, 1, \ldots, n-1$

Lemma 1. (*Erdös and Szekeres, 1935*.) Given a sequence of n distinct numbers $x_1, x_2, \ldots, x_n$, we can decompose this set into k increasing subsequences $\ell_1, \ell_2, \ldots, \ell_k$ such that:

- There is a decreasing subsequence of length k;
- If x_i belongs to ℓ_a then for all $j > i$ if $x_j < x_i$ then x_j belongs to ℓ_b and $b > a$.

Release dates: Lower bound

Theorem. The PoR in the presence of release dates is $\Omega(\frac{\log n}{\log \log n})$

Instance: $w_j = 1$, $p_j = 2^j$, $r_{n-1-j} = \sum_{i>j} 2^i = \sum_{i>j} p_i$, $j = 0, 1, \ldots, n-1$

Lemma 1. (*Erdös and Szekeres, 1935*.) Given a sequence of n distinct numbers $x_1, x_2, \ldots, x_n$, we can decompose this set into k increasing subsequences $\ell_1, \ell_2, \ldots, \ell_k$ such that:

- There is a decreasing subsequence of length k;
- If x_i belongs to ℓ_a then for all $j > i$ if $x_j < x_i$ then x_j belongs to ℓ_b and $b > a$.

Lemma 2. Universal schedule with decreasing subsequence ℓ has ratio at least $|\ell|$.

Release dates: Lower bound

Theorem. The PoR in the presence of release dates is $\Omega(\frac{\log n}{\log \log n})$

Instance: $w_j = 1, p_j = 2^j, r_{n-1-j} = \sum_{i>j} 2^i = \sum_{i>j} p_i, j = 0, 1, \ldots, n-1$

Lemma 1. (*Erdös and Szekeres, 1935.*) Given a sequence of n distinct numbers $x_1, x_2, \ldots, x_n$, we can decompose this set into k increasing subsequences $\ell_1, \ell_2, \ldots, \ell_k$ such that:

- There is a decreasing subsequence of length k;
- If x_i belongs to ℓ_a then for all $j > i$ if $x_j < x_i$ then x_j belongs to ℓ_b and $b > a$.

Lemma 2. Universal schedule with decreasing subsequence ℓ has ratio at least $|\ell|$.

Let j first job in ℓ. Then breakdown at $[r_j, r_0]$ and $[r_0 + 2^j - 1, L]$. At time r_0 all jobs are released. At time r_0 (end of first breakdown), job j starts processing and is not completed by $r_0 + 2^j - 1$. Hence all jobs of ℓ are uncompleted, whereas an optimal schedule can complete all jobs except j. Choose L large enough.

Release dates: Lower bound

Theorem. The PoR in the presence of release dates is $\Omega(\frac{\log n}{\log \log n})$

Instance: $w_j = 1$, $p_j = 2^j$, $r_{n-1-j} = \sum_{i>j} 2^i = \sum_{i>j} p_i$, $j = 0, 1, \ldots, n-1$

Lemma 1. (*Erdös and Szekeres, 1935*.) Given a sequence of n distinct numbers $x_1, x_2, \ldots, x_n$, we can decompose this set into k increasing subsequences $\ell_1, \ell_2, \ldots, \ell_k$ such that:

- There is a decreasing subsequence of length k;
- If x_i belongs to ℓ_a then for all $j > i$ if $x_j < x_i$ then x_j belongs to ℓ_b and $b > a$.

Lemma 2. Universal schedule with decreasing subsequence ℓ has ratio at least $|\ell|$.

Lemma 3. Universal schedule with decomposition $\ell_1, \ell_2, \ldots, \ell_k$ has ratio at least $\frac{|\ell_i|+|\ell_{i-1}|+\cdots+|\ell_1|}{1+|\ell_{i-1}|+\cdots+|\ell_1|}$, $i = 1, \ldots, k$.

Release dates: Lower bound

Theorem. The PoR in the presence of release dates is $\Omega(\frac{\log n}{\log \log n})$

Instance: $w_j = 1$, $p_j = 2^j$, $r_{n-1-j} = \sum_{i>j} 2^i = \sum_{i>j} p_i$, $j = 0, 1, \ldots, n-1$

Lemma 1. (*Erdös and Szekeres, 1935.*) Given a sequence of n distinct numbers $x_1, x_2, \ldots, x_n$, we can decompose this set into k increasing subsequences $\ell_1, \ell_2, \ldots, \ell_k$ such that:

- There is a decreasing subsequence of length k;
- If x_i belongs to ℓ_a then for all $j > i$ if $x_j < x_i$ then x_j belongs to ℓ_b and $b > a$.

Lemma 2. Universal schedule with decreasing subsequence ℓ has ratio at least $|\ell|$.

Lemma 3. Universal schedule with decomposition $\ell_1, \ell_2, \ldots, \ell_k$ has ratio at least $\frac{|\ell_i|+|\ell_{i-1}|+\cdots+|\ell_1|}{1+|\ell_{i-1}|+\cdots+|\ell_1|}$, $i = 1, \ldots, k$.

For all $j \in \ell_i$ breakdown $[r_j, r_j + \epsilon]$ and for all $j \in \ell_{i+1}, \ldots, \ell_k$ breakdown $[r_j, r_j + 2^j] = [r_j, r_j + p_j]$. At time $2^n - 1$ all jobs in $\ell_i, \ell_{i+1}, \ldots, \ell_k$ are uncompleted. Compare with a schedule that leaves only the last job of ℓ_i and all jobs in $\ell_{i+1}, \ldots, \ell_k$ uncompleted. Thus a breakdown during $[2^n - 1, L]$ for L large enough yields the proof.

Release dates: Lower bound

Theorem. The PoR in the presence of release dates is $\Omega(\frac{\log n}{\log \log n})$

Instance: $w_j = 1$, $p_j = 2^j$, $r_{n-1-j} = \sum_{i>j} 2^i = \sum_{i>j} p_i$, $j = 0, 1, \ldots, n-1$

Lemma 1. (*Erdös and Szekeres, 1935*.) Given a sequence of n distinct numbers $x_1, x_2, \ldots, x_n$, we can decompose this set into k increasing subsequences $\ell_1, \ell_2, \ldots, \ell_k$ such that:

- There is a decreasing subsequence of length k;
- If x_i belongs to ℓ_a then for all $j > i$ if $x_j < x_i$ then x_j belongs to ℓ_b and $b > a$.

Lemma 2. Universal schedule with decreasing subsequence ℓ has ratio at least $|\ell|$.

Lemma 3. Universal schedule with decomposition $\ell_1, \ell_2, \ldots, \ell_k$ has ratio at least

$\frac{|\ell_i|+|\ell_{i-1}|+\cdots+|\ell_1|}{1+|\ell_{i-1}|+\cdots+|\ell_1|}$, $i = 1, \ldots, k$.

Proof of Theorem: Let α performance guarantee. Using Lemma 3, any $|\ell_i| \leq \alpha^{k-i+1}$ (by induction).
$n = \sum_{i=1}^{k} |\ell_i| \leq \sum_{i=1}^{k} \alpha^{k-i+1} \leq \alpha^{k+1}$.
By Lemma 2 $k \leq \alpha$. Therefore $\log n = O(\alpha \log \alpha)$ hence the lower bound.

Release dates

Positive result:

- Worst-case ratio of 5 (tight) if $w_j/p_j = \beta$, for all j.

Release dates

Positive result:

- Worst-case ratio of 5 (tight) if $w_j/p_j = \beta$, for all j.
- For $w_j/p_j = \beta$, for all j the price of robustness at least 3.

Other results

- The algorithm can be extended to special cases of jobs with precedence constraints
- The first constant ratio algorithm for scheduling in the presence of unavailable periods
- If there is only one unavailability period (speed 0) then there exists a FPTAS

  Independently achieved by Kellerer and Strusevich, but ours has smaller running time

Open problems

- Release dates:
  - Find a universal schedule that has performance ratio $O(\log n)$.
- Multiple machines:
  - Consider two machines; if both machines can break simultaneously then it is easy to see that there is no robust approximation algorithm unless P=NP (reduction from partition)
  - Good robust model for multiple machines; e.g. (at most) 1 machine breaking/slowing down at a time

Open problems

- Expensive intervals:
  - Some time intervals are freely available and for all other (expensive) time units can be made available at cost c per time unit. Still minimising $\sum w_j C_j$
  - Without release dates easy, but not trivial
  - With release time and $p_j = 1$ is easy, but not trivial
  - With release times and general processing time not known (guess that it is easy without weights)