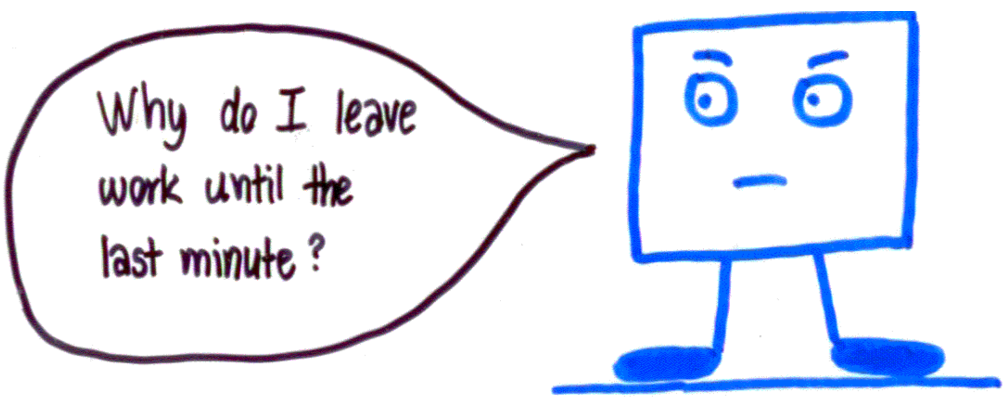
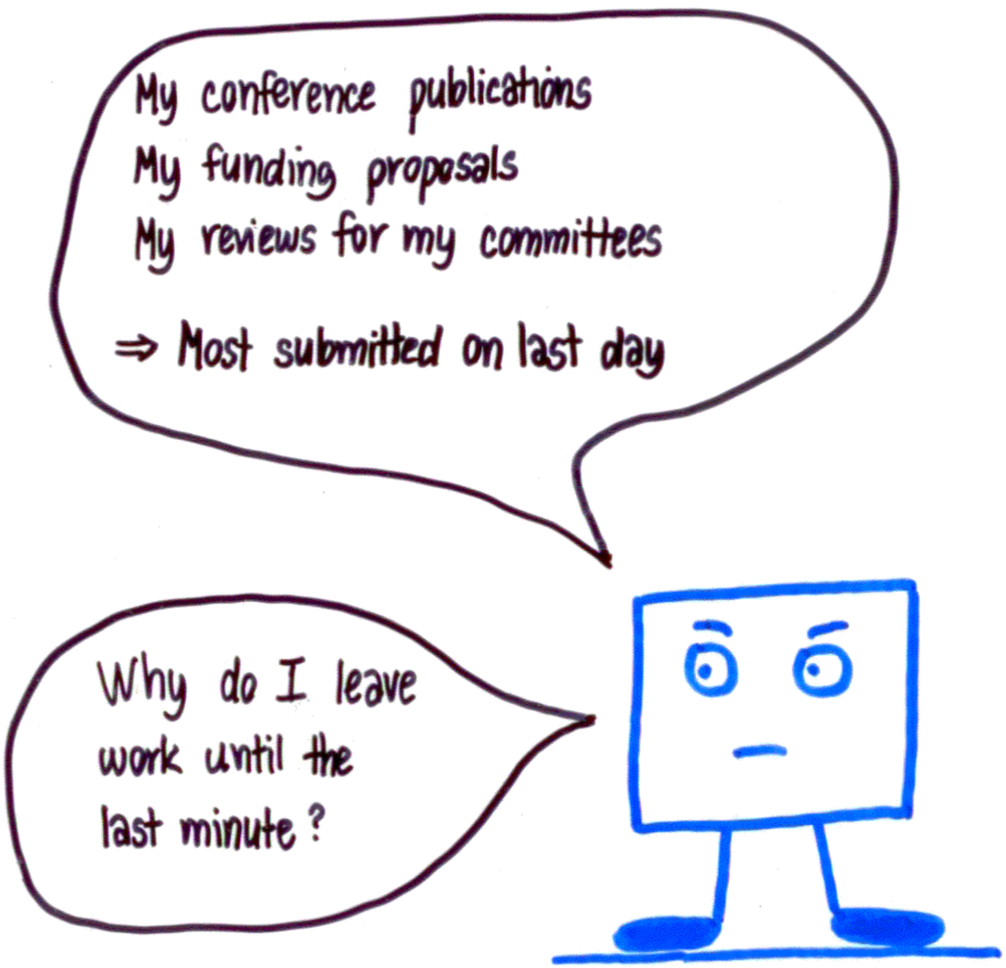
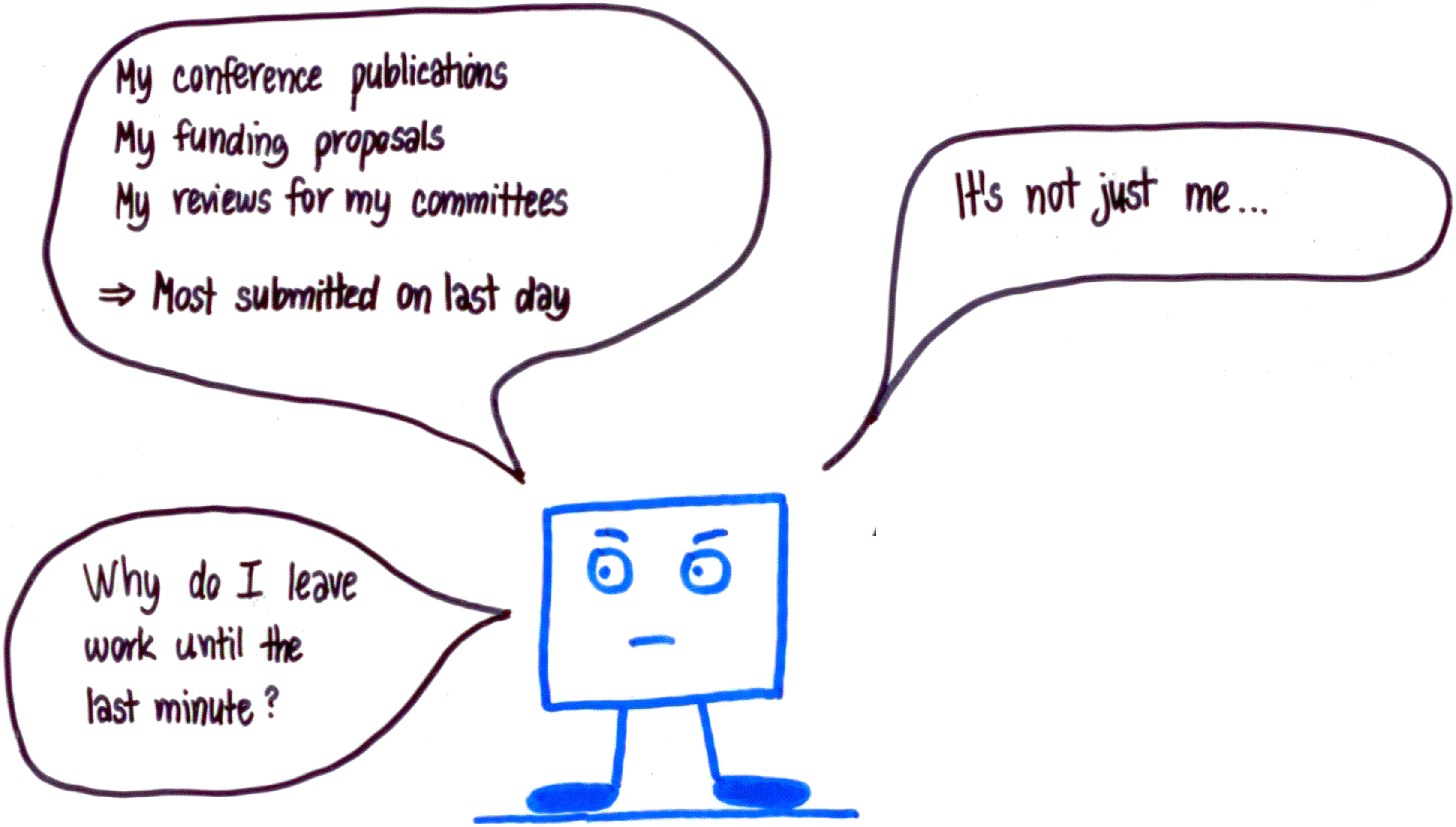
Michael Being Introspective



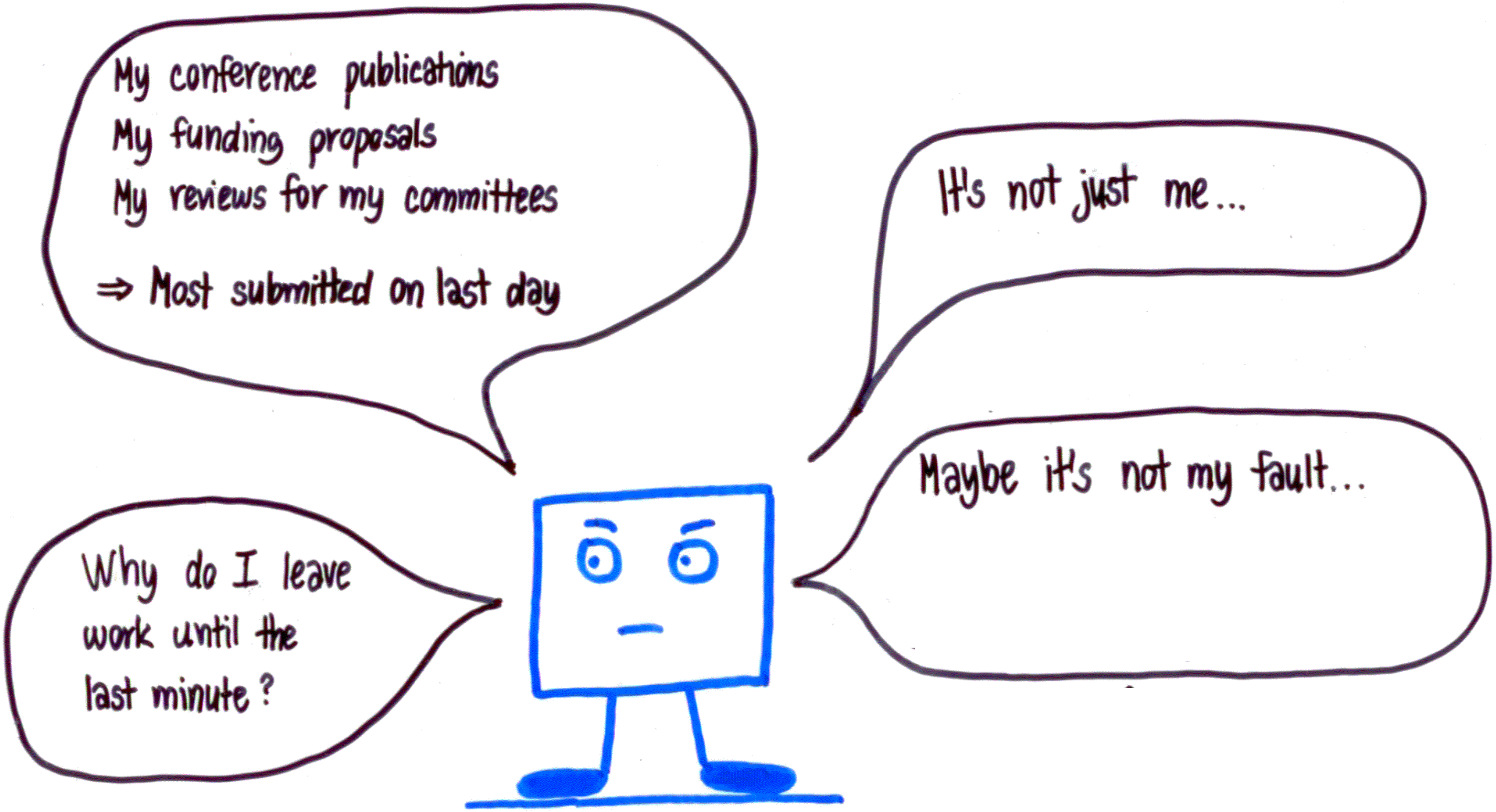
Michael Being Introspective



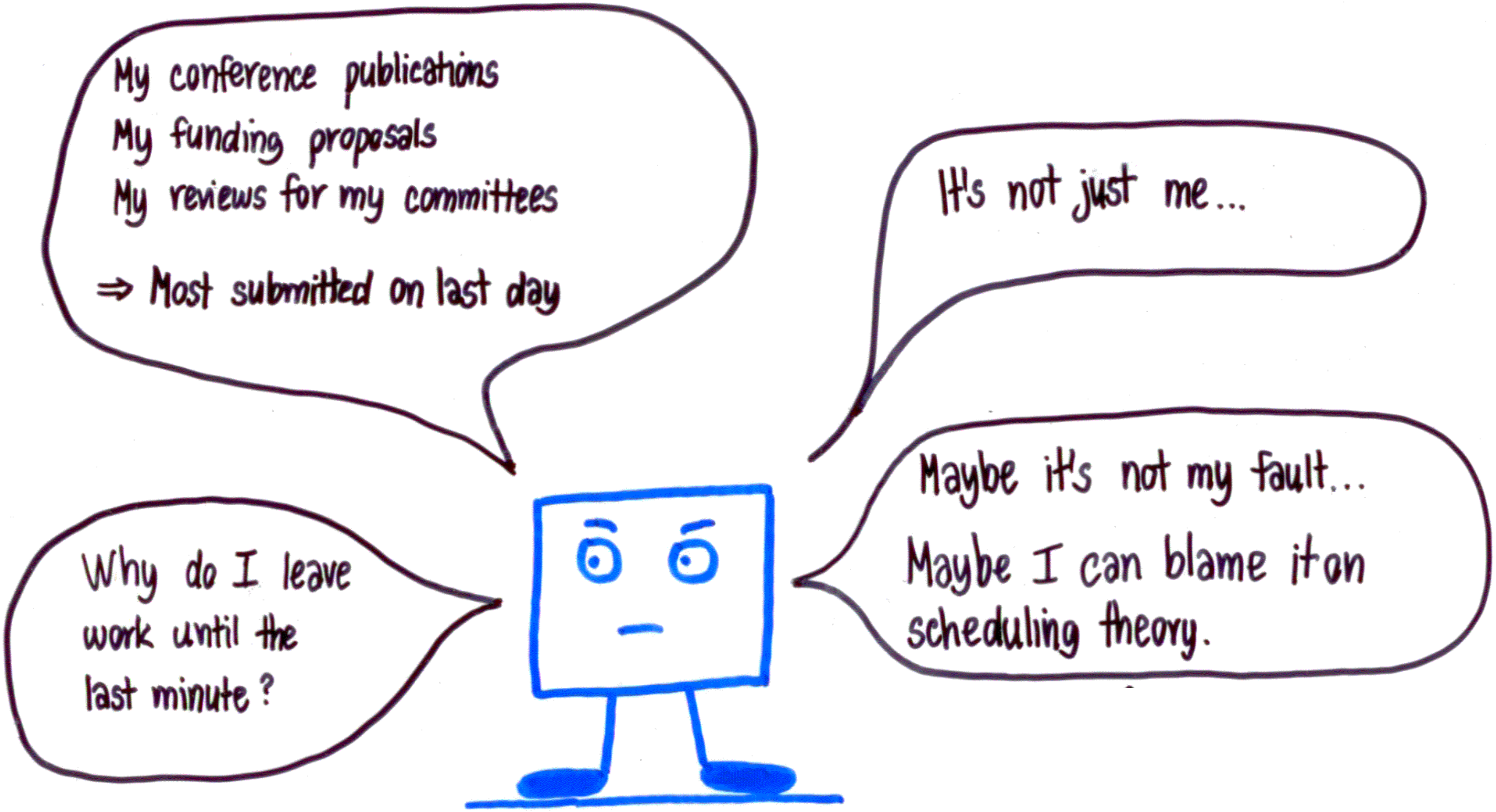
Michael Being Introspective



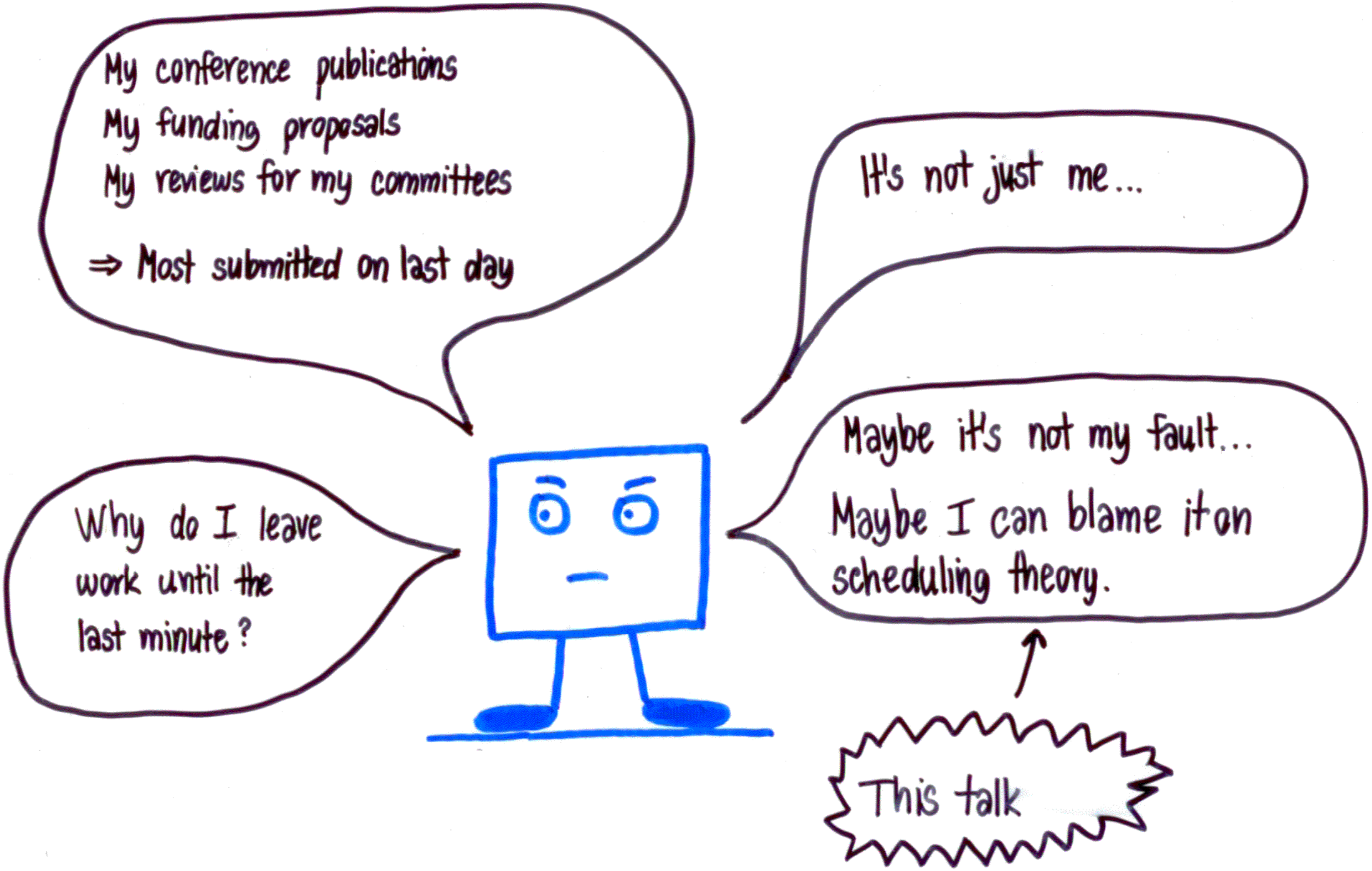
Michael Being Introspective



Michael Being Introspective



Michael Being Introspective



Translation of Procrastination

- Common English word. Suggests that you will get the thing done, but at the last minute.
  - You guarantee yourself pain and possible failure. Until the pain, all you have is a vague sense of worry.
  - Mothers berate children: "Don't procrastinate"
- Couldn't find agreement of good translations in French and Polish. Why?

आज करे सो काल कर
काल करे सो परसों
इतनी भी जल्दी क्या
दैर पड़ा है बरसों

कबीर रबीक

Famous Indian Poet

Advantage of Procrastination

The closer it is to the deadline, the faster we work and the less time a job takes.

This is a scheduling problem.

The Procrastination Scheduling Problem

Michael Bender
Stony Brook

Raphael Clifford
Bristol

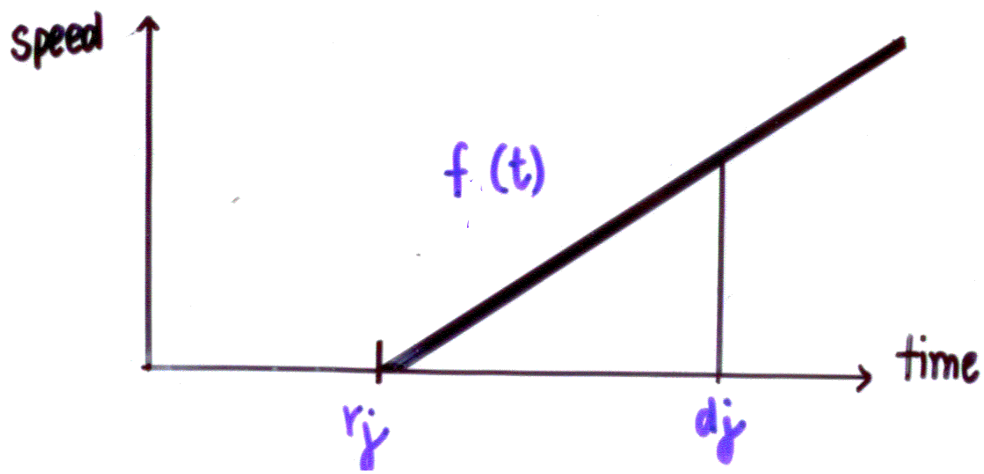
Kostas Tsichlas
Patras

Procrastination Scheduling Problem

Jobs $1 \ldots n$.

Release times $r_1 \ldots r_n$ and deadlines $d_1 \ldots d_r$.

Preemption

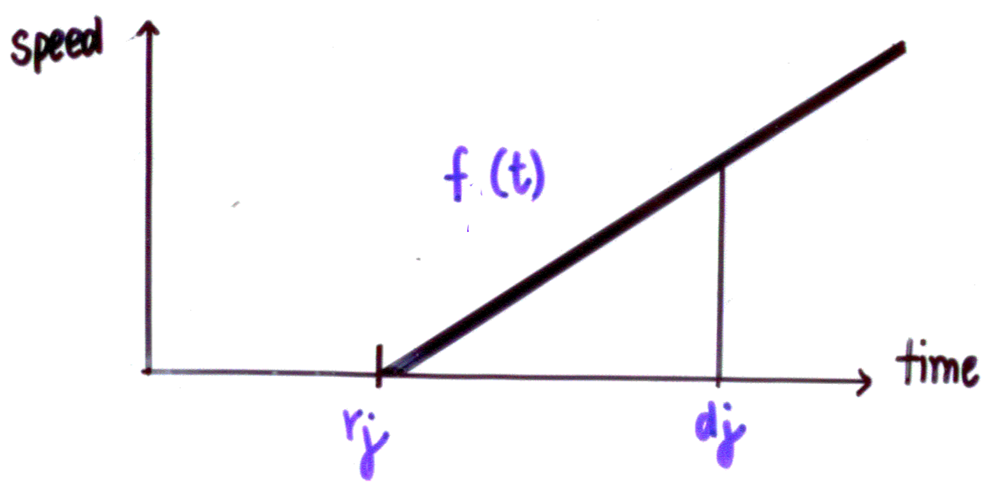


Procrastination Scheduling Problem

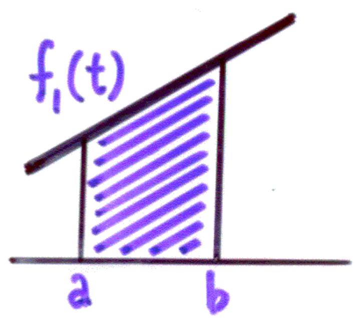
Jobs $1 \ldots n$.

Release times $r_1 \ldots r_n$ and deadlines $d_1 \ldots d_r$.

Preemption



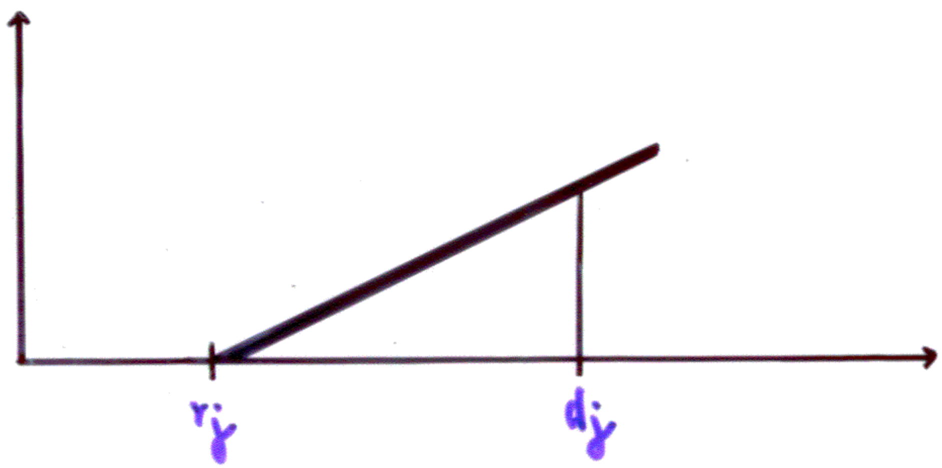
Work on job j in $[a,b]$ is $\int_{t=a}^{b} dt\, f_j(t)$.



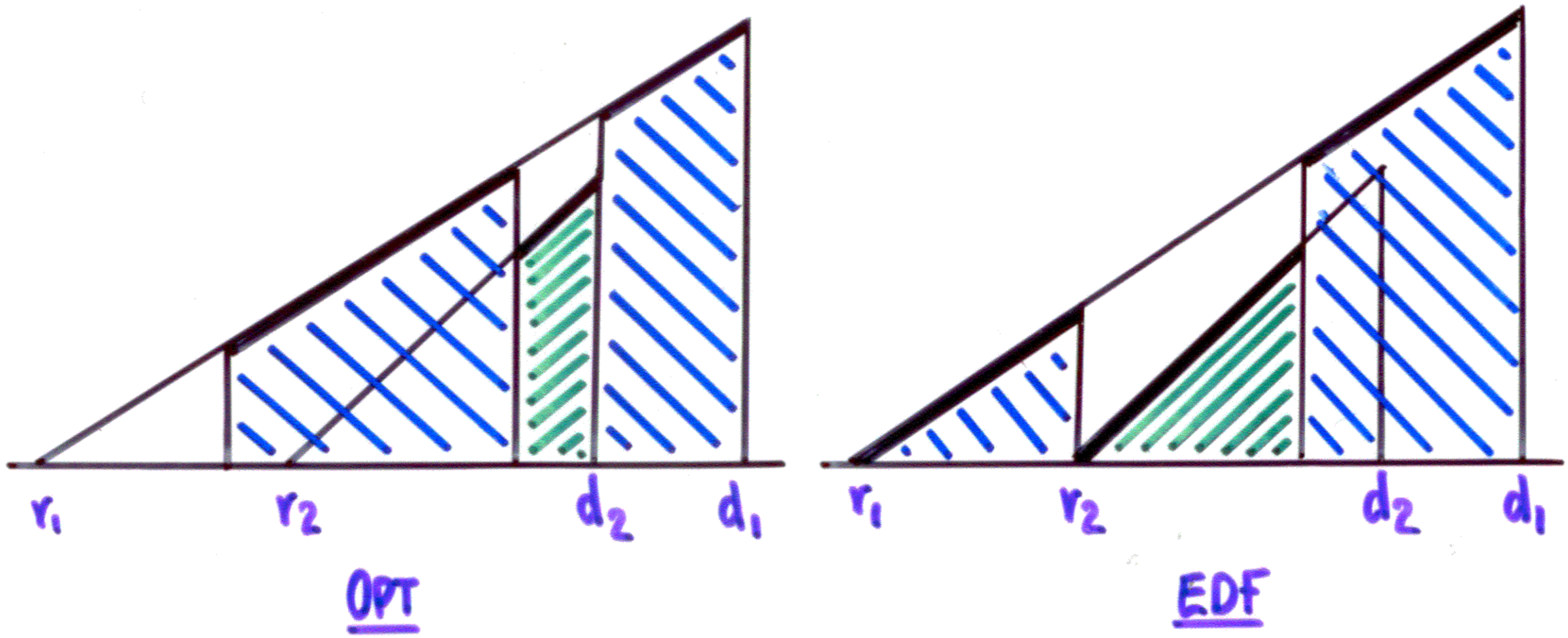
<u>Goal</u>: Find the feasible schedule that minimizes running time.

This Talk

Linear speed functions $f_j(t) = m_j(t - r_j)$



Why Earliest Deadline First (EDF) is Not Optimal



EDF is not optimal.

Better to run job 2 near its deadline for maximal efficiency

Results

- OPT offline policy for procrastinators.
- Difficulty of online scheduling.

→ Online scheduler is harder for procrastinators than nonprocrastinators (true in math and real life).

Results

- OPT offline policy for procrastinators.
- Difficulty of online scheduling.

  → Online scheduler is harder for procrastinators than nonprocrastinators (true in math and real life).

- Numerical/computational difficulty of procrastination scheduling.

  → Not known to be in NP, despite simple policy.

Results

- OPT offline policy for procrastinators.
- Difficulty of online scheduling.

  → Online scheduler is harder for procrastinators than nonprocrastinators (true in math and real life).

- Numerical/computational difficulty of procrastination scheduling.

  → Not known to be in NP, despite simple policy.

- Online algorithms for ***max-interval-stretch.*** Metric for procrastinator that is frequently late, but never by much.
- Analysis of common algs of procrastinators:
  - "Hit the highest nail" alg is has unbounded max-interval-stretch
  - "Thrashing alg" is O(1) competitive

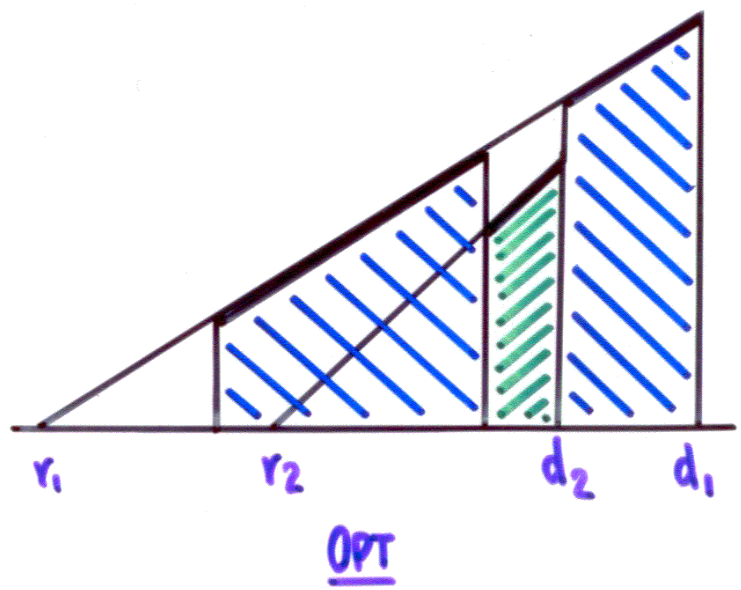
Related Work

- Time-dependent shortest paths [Orda, Rom 90] and network flows [Fleischer & Skutella 02 03].
- Scheduling with time-dependent tasks [Alidaee & Womer 99] [Bachman, Janiak, Kovalyov 02] [Gawiejnovicz, Kurc, Pankowska 02][Gawiejnowicz & Pankowska 95]
  No preemption changes problem completely. (E.g., requires infinite speeds.)
- Speed scaling [Irani Pruhs] [Bunde 06] [etc].
- Lazy Bureaucrat Scheduling Problem [Arkin, Bender,Mitchell, Skiena 99, 03].

Optimal Policy: Latest Release Time Backwards

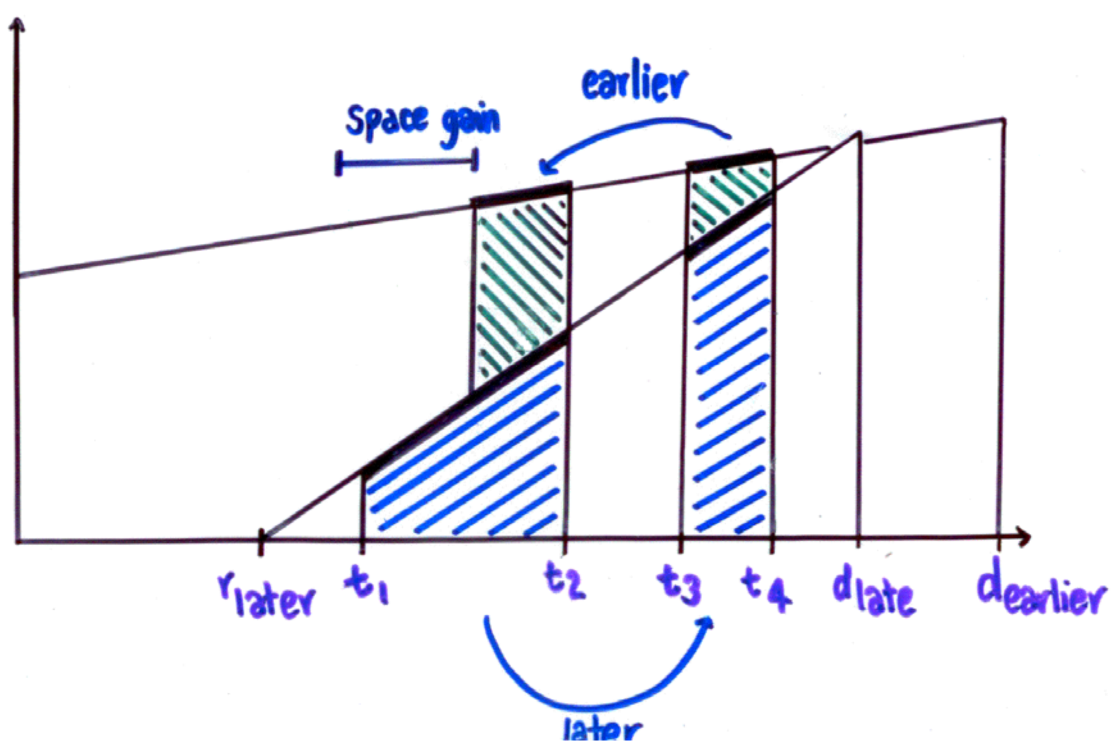
Allocating backwards in time, run the job with the latest release time.

(i.e.: Earliest deadline first with time running backwards)



Exchange Argument of LRTB

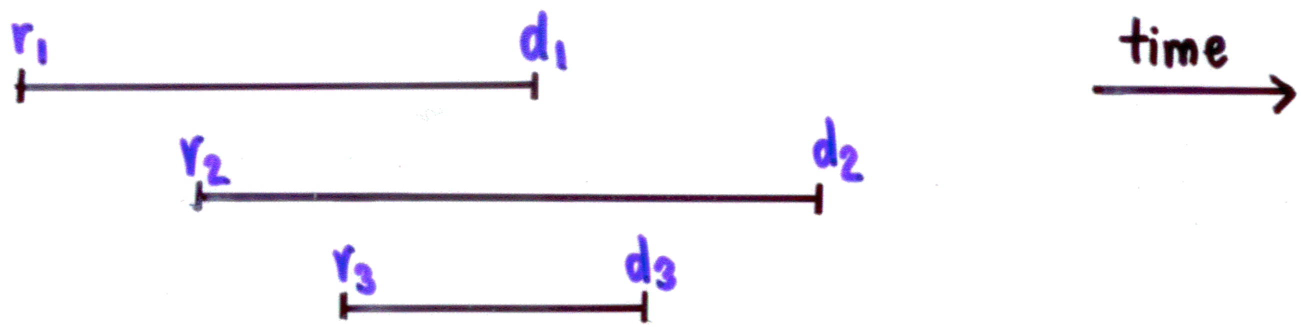
- Assume by contradiction that ∃ scheduler S with smaller total processing time.
- Then S executes some earlier job during some $[t_3,t_4]$ instead of later job, which is executed during $[t_1,t_2]$.
- Intervals $[t_3,t_4]$ $[t_1,t_2]$ defined s.t. $\int_{t=t_1}^{t_2} dt\, f_{later}(t) = \int_{t=t_3}^{t_4} dt\, f_{later}(t)$



LRTB (Latest Release-Time Backwards)

Remark: EDF for traditional scheduling works the same regardless of whether times runs forwards or backwards.

(If time runs backwards, switch roles of release times and deadlines.)



But not for the procrastinator!

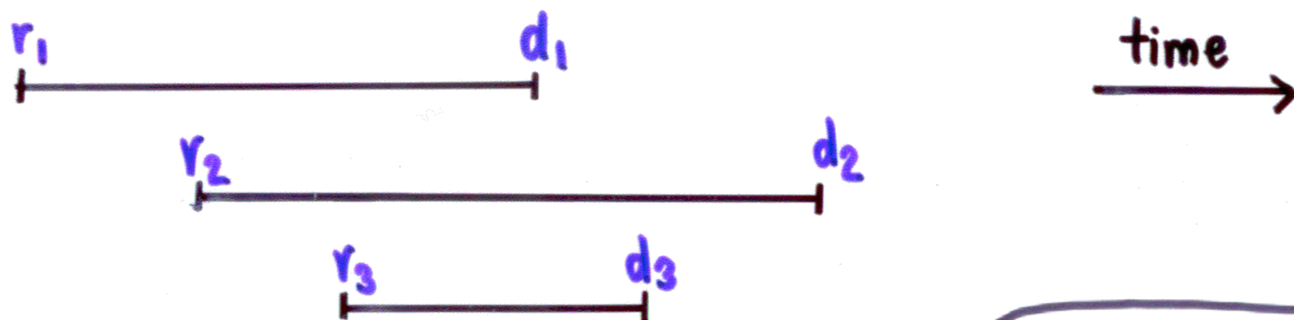
EDF only works when time flows backwards

That's why the procrastinator says:

LRTB (Latest Release-Time Backwards)

Remark: EDF for traditional scheduling works the same regardless of whether times runs forwards or backwards.

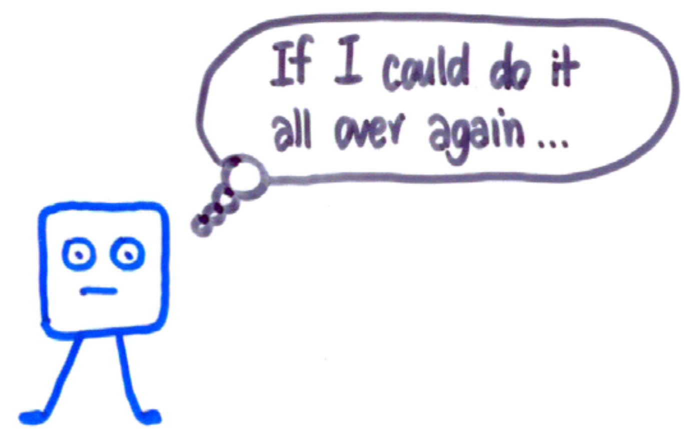
(If time runs backwards, switch roles of release times and deadlines.)



But not for the procrastinator!

EDF only works when time flows backwards

That's why the procrastinator says:



Complexity: LRTB (Latest Release-Time Backwards)

Elegant scheduling policy but not known to be in NP:

Numerical/computational difficulty with square roots.

Complexity: LRTB (Latest Release-Time Backwards)

Elegant scheduling policy but not known to be in NP:

Numerical/computational difficulty with square roots.

Sum-of-Square-Roots Problem: For integers $x_1, \ldots, x_n$, and I, is

$$\sum_{i=1}^{n} \sqrt{x_i} < I \; ?$$

Not known to be in NP.

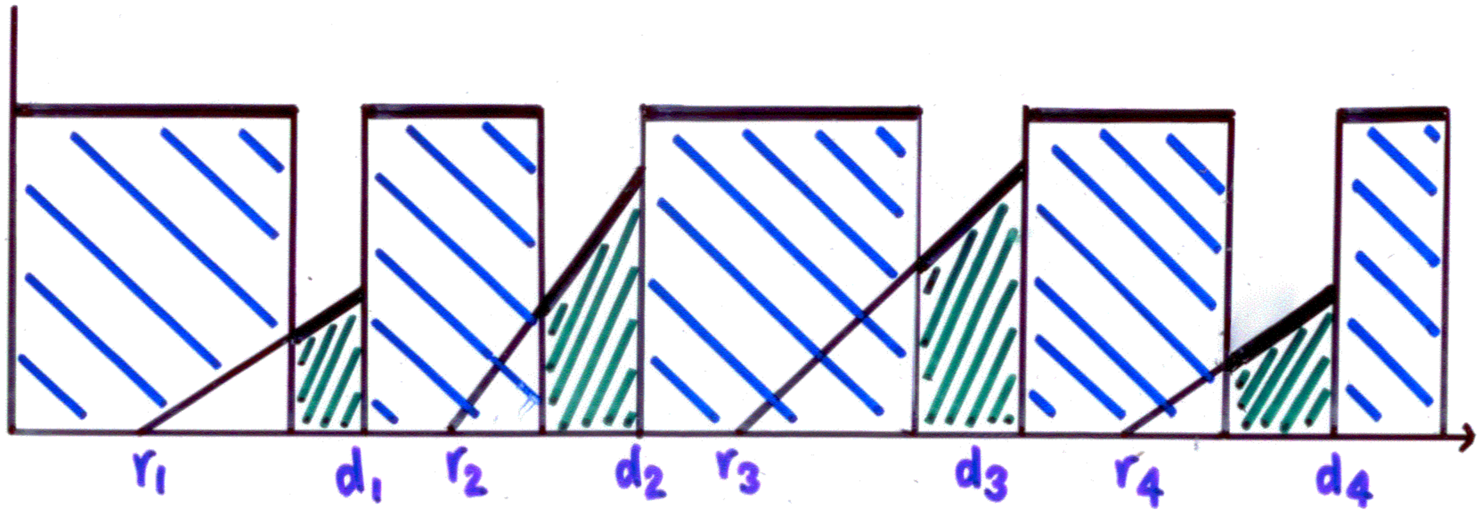
$\Rightarrow$ Euclidean TSP and shortest path not known to be in NP.

Procrastination scheduling: speeds introduce square roots.

Illustration of Numerical Difficulty.

n procrastinating jobs

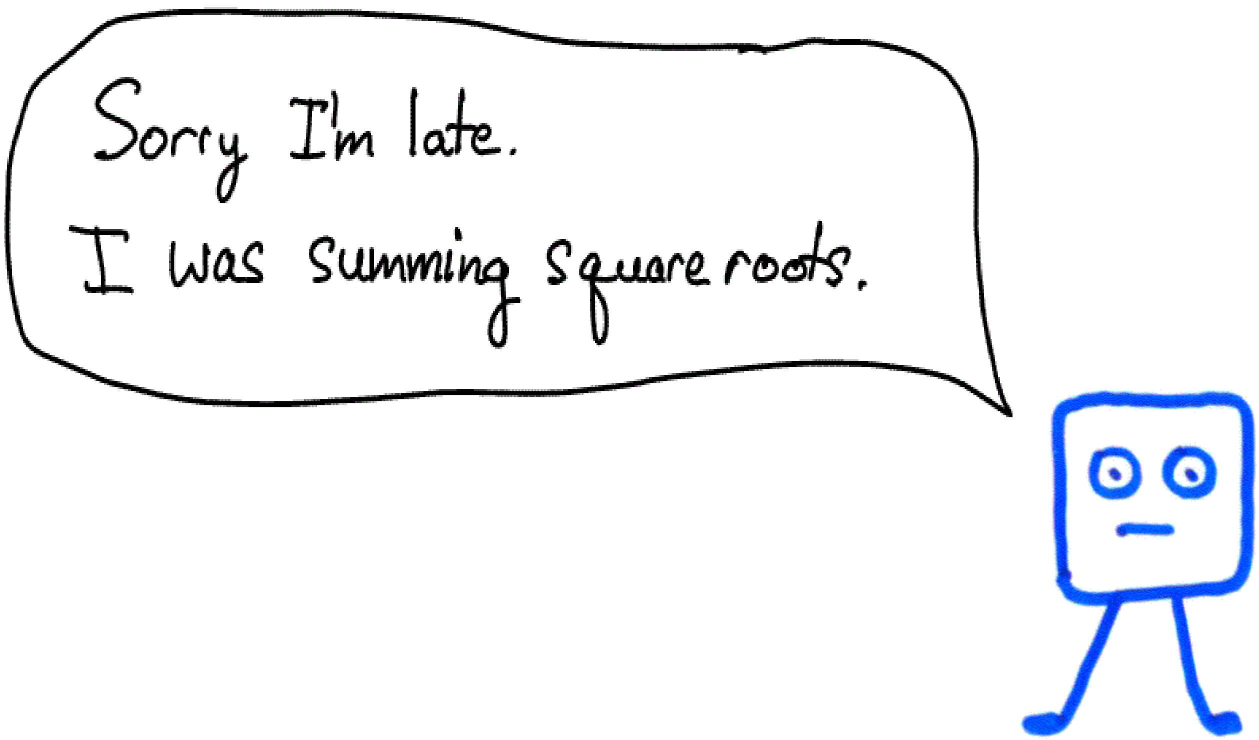
1 non-procrastinating job



OPT solution: run procrastinating jobs at deadlines.

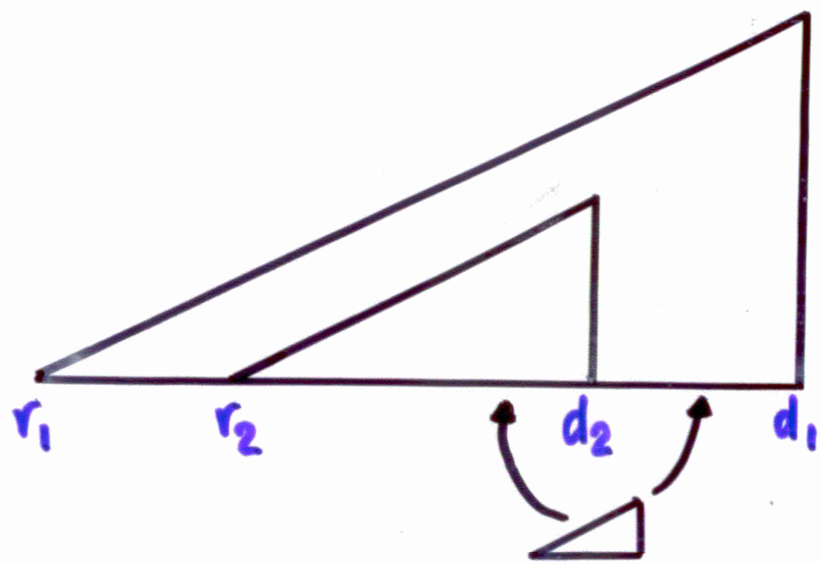
Calculating sum of processing times means summing square roots.

Life is Difficult for Procrastinators



ONLINE PROCRASTINATION SCHEDULING

Theorem (of little hope): For any online algorithm, there is a feasible job stream on which that algorithm misses deadlines.

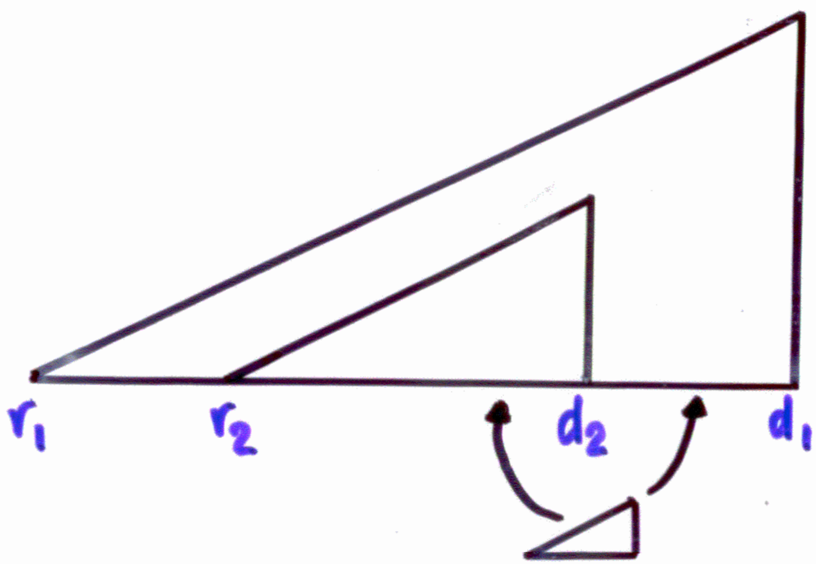


Procrastinator at time r_2: do I execute job 1 or 2?

May depend on location of job 3.

ONLINE PROCRASTINATION SCHEDULING

Theorem (of little hope): For any online algorithm, there is a feasible job stream on which that algorithm misses deadlines.



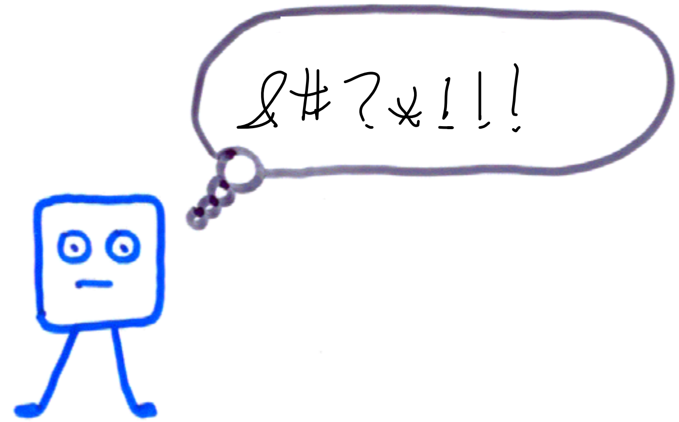
Procrastinator at time r_2: do I execute job 1 or 2?

May depend on location of job 3.

Explains why procrastinators may have a harder time than non-procrastinators.

Behavior of Many Procrastinators

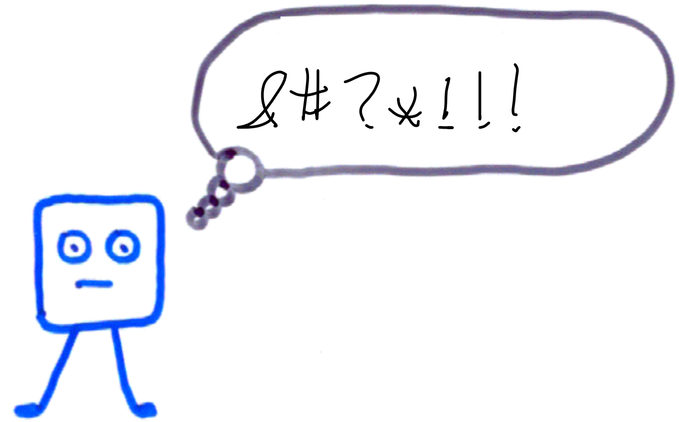
Late on everything but not by too much



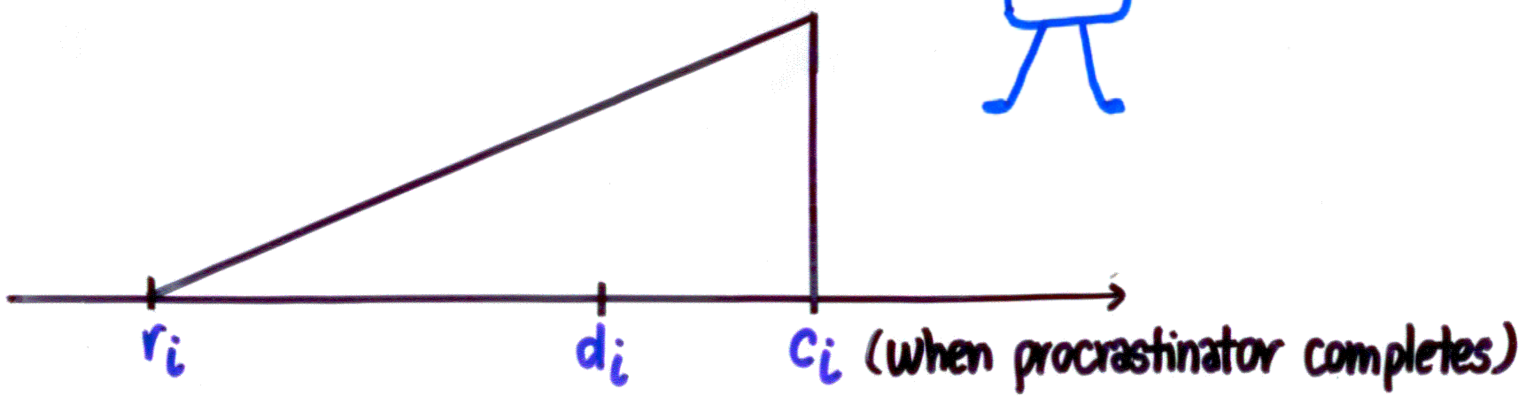
Def: Interval stretch.

Behavior of Many Procrastinators

Late on everything but not by too much



__Def__: Interval stretch.



$$\text{Interval stretch} = \frac{\text{time in system (flowtime)}}{\text{actual interval of job}} = \frac{c_i - r_i}{d_i - r_i}$$

Many procrastinators try to minimize __max__-interval stretch.

Online Scheduling for Procrastinators

Not surprisingly, traditional algorithms for non-procrastinators do not work well for procrastinators.

SRPT and EDF generate schedules that have unbounded max internal stretch.

Online Scheduling for Procrastinators

Not surprisingly, traditional algorithms for non-procrastinators do not work well for procrastinators.

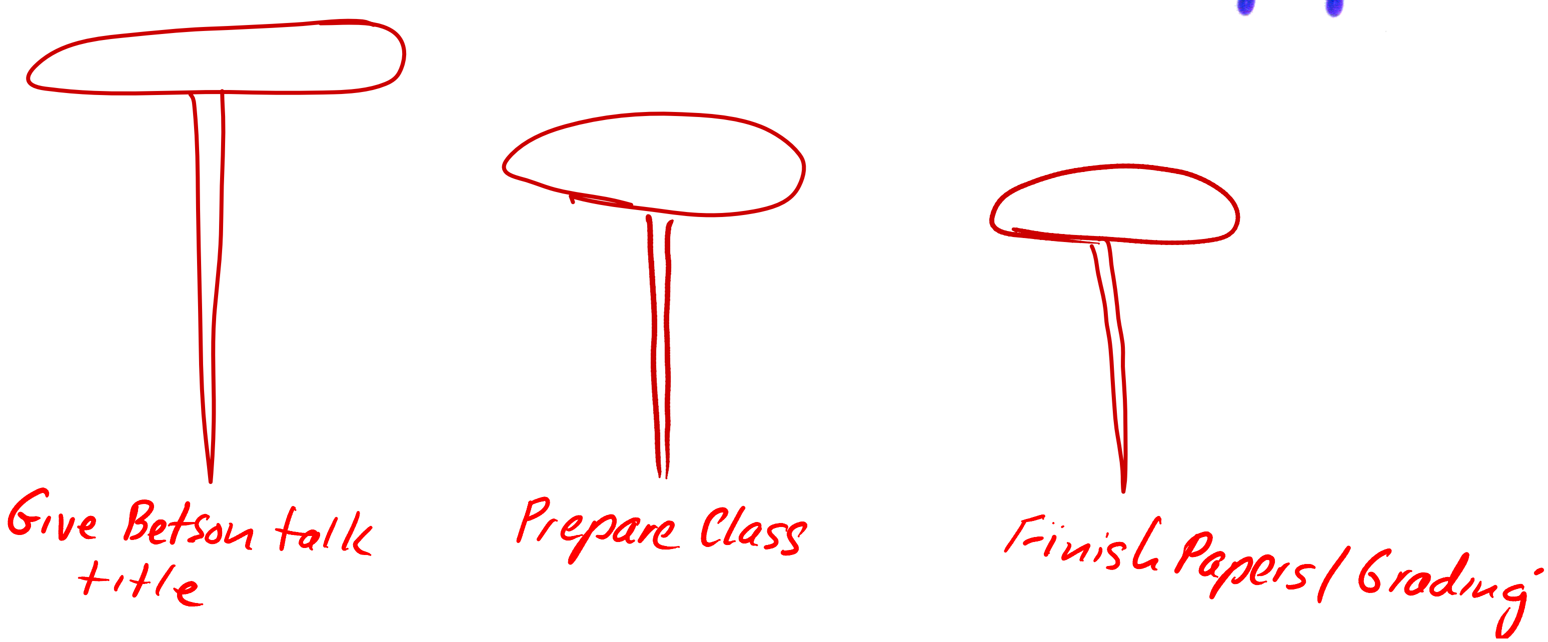
SRPT and EDF generate schedules that have unbounded internal stretch.

But what about algs that procrastinators use?

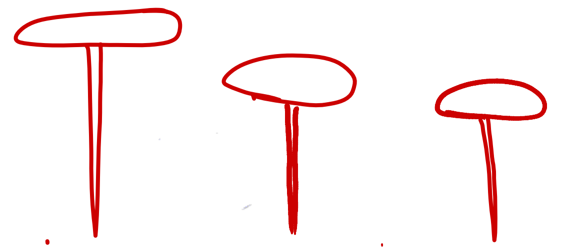
"Hit the Highest Nail"

Many procrastinators work on the job that is most urgent, i.e., that has the largest interval stretch so far.

At time t, execute job j in system that maximizes $\dfrac{t - r_j}{d_j - r_j}$.



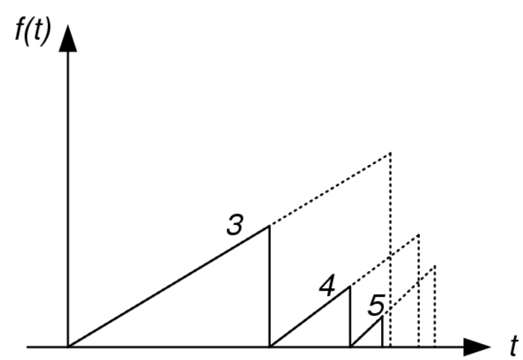
"Hit the Highest Nail"



Many procrastinators work on the job that is most urgent, i.e., that has the largest interval stretch so far.

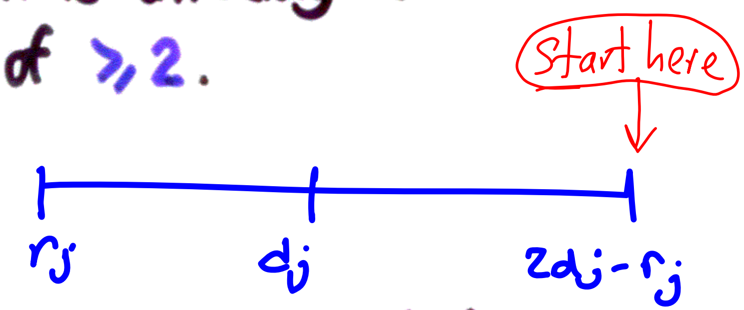
At time t_1, execute job j in system that maximizes $\frac{t - r_j}{d_j - r_j}$.

Common approach still leads to unbounded max-interval stretch.



THRASHING: O(1)-Competitive Max-Interval Stretch

- Don't work on anything until it is already late and has max interval stretch of ≥ 2.



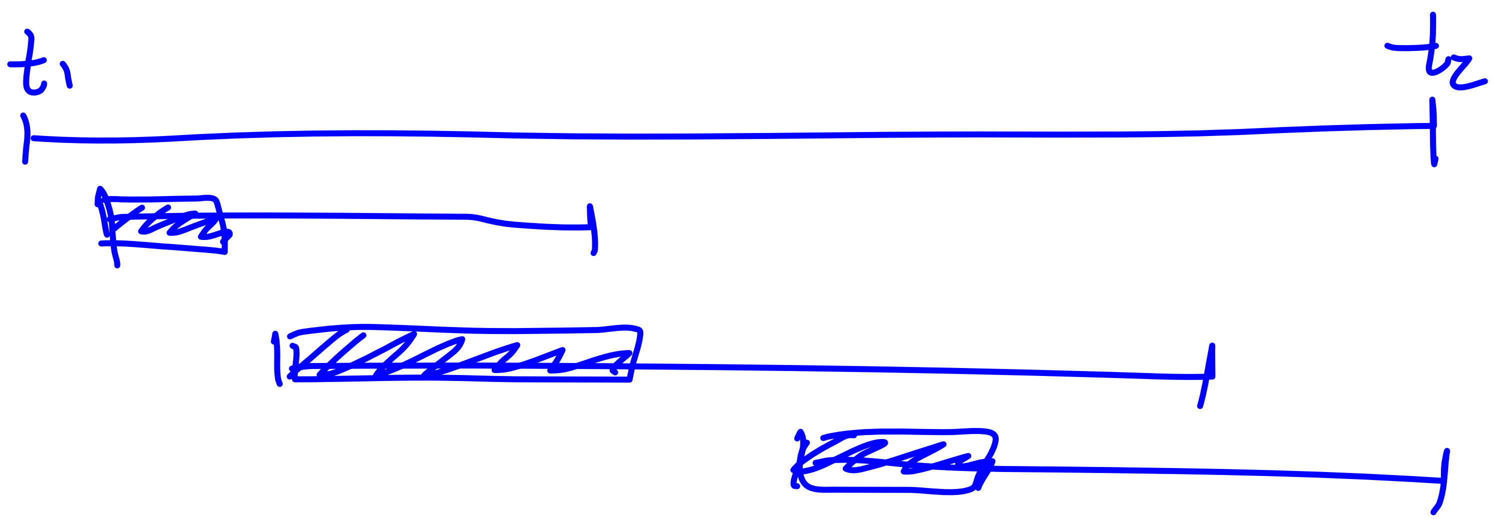
- Then work LIFO
  i.e., work on the last job to arrive on your desk.

<u>Theorem</u>: This algorithm is O(1) competitive for max-internal stretch.

Procrastinator never runs more than O(1) faster than OPT.

Simple Lemma for Thrashing Algorithm

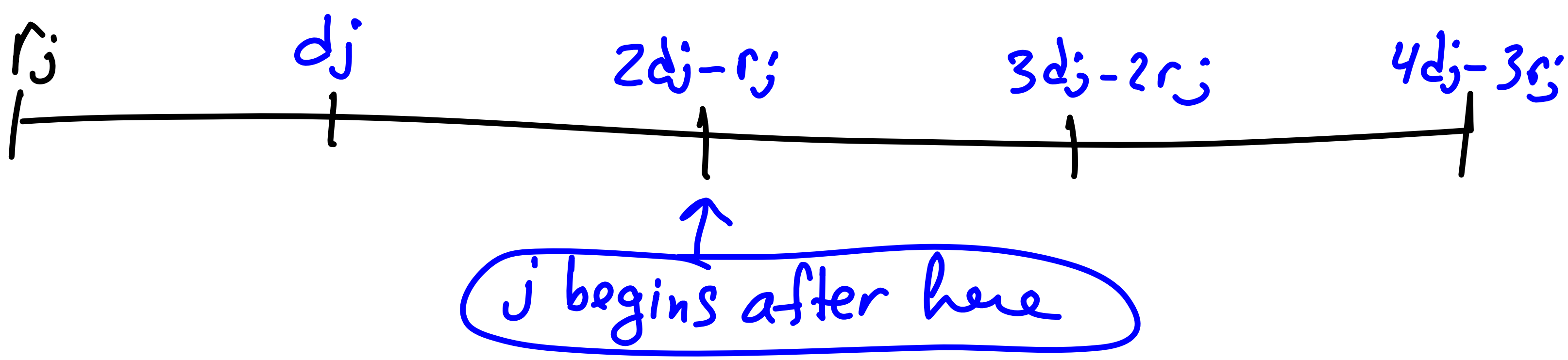
- Given a feasible set of jobs whose intervals are in $[t_1, t_2]$, the thrashing algorithm completes all jobs using $(t_1 - t_2)/2$ time.



Thrashing Algorithm is 4-Competitive

- Jobs arriving before r_j cannot block j.
- Total work OPT schedules in $[r_j, 4(d_j-r_j) + r_j]$ fits in at most $2(d_j-r_j)$ steps.

→ j must complete.



Results

- OPT offline policy for procrastinators.
- Difficulty of online scheduling.

→ Online scheduler is harder for procrastinators than nonprocrastinators (true in math and real life).

- Numerical/computational difficulty of procrastination scheduling.

→ Not known to be in NP, despite simple policy.

- Online algorithms for ***max-interval-stretch.*** Metric for procrastinator that is frequently late, but never by much.
- Analysis of common algs of procrastinators:
  - "Hit the highest nail" alg is has unbounded max-interval-stretch
  - "Thrashing alg" is O(1) competitive

Problems to Think About Some Time in the Future

- Tighter bounds
- Other speed functions
- Procrastinators working in parallel
- Other metrics
- Closer modeling of human behavior and applications
- Connect to the many applications where late processors operate at unsustainable rates

CONCLUSION

$(\text{CONCLUSION})^{-1}$ "Conclusion in verse"

$(\text{Conclusion})^{-1}$ "Conclusion in verse"

When you wait 'til it's almost too late,
then you run at a much faster rate.
You accomplish more work
When you're going berserk
It's innate that you procrastinate.

$(\text{CONCLUSION})^{-1}$ "Conclusion in verse"

When you wait 'til it's almost too late,
then you run at a much faster rate.
You accomplish more work
When you're going berserk
It's innate that you procrastinate.