Three models for scheduling under explorable uncertainty

Christoph Dürr
Sorbonne Université, CNRS

schedulingseminar.com
December 2021

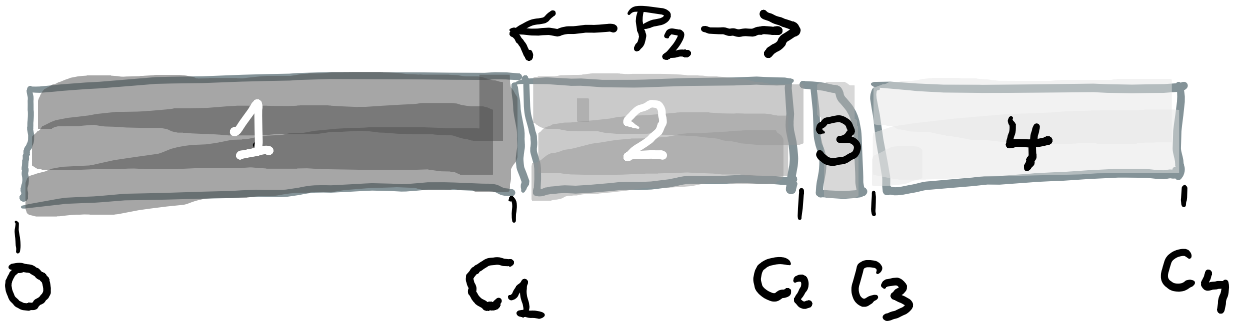
The standard model

Single machine, n jobs, each job j has processing time p_j, and priority weight w_j.

Objective: minimize $\Sigma w_j C_j$,
where C_j=completion time of job j

Optimum: schedule in order of decreasing Smith-ratio

$$\frac{w_1}{p_1} \geq \cdots \geq \frac{w_n}{p_n}$$



I know how to optimize your schedule, just give me the processing times and weights.

Hmm, we don't have this information right now.

Motivation: serving patients in an emergency departement

Model 1

Levi, Magnanti, Shaposhnik, Scheduling with Testing, Management Science, 2019

Notations changed for the talk

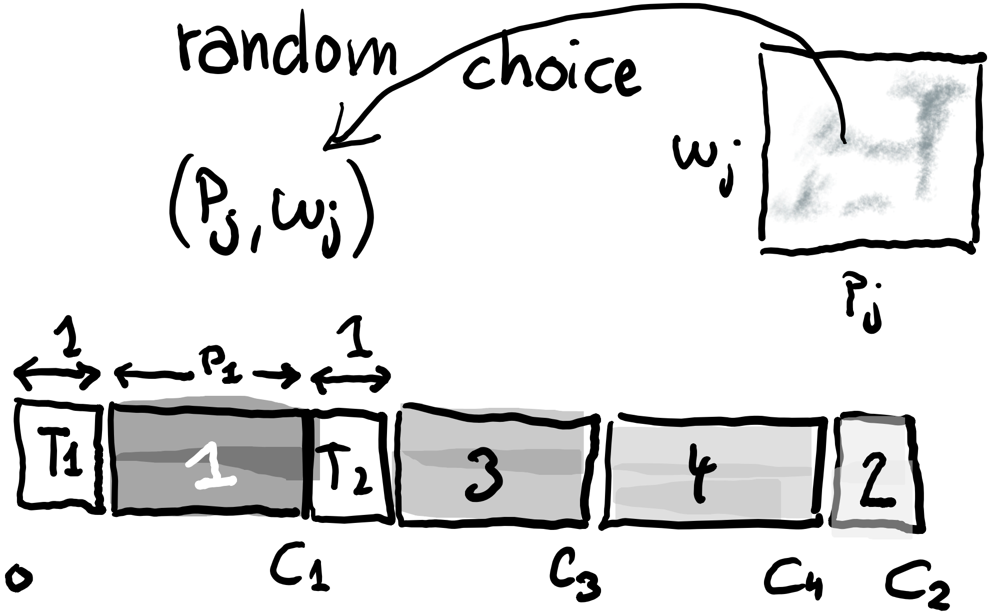
Single machine, n jobs, each job j has processing time pj, and priority weight wj.

(p_j, w_j) are randomly chosen from a joint distribution, identical for each j.

Initially only the distribution is known, not the actual job characteristics.

Algorithm can do a test for a specific job j, revealing p_j, w_j, it occupies 1 time unit on the schedule.

Objective: minimize $\mathbf{E}[\Sigma w_j C_j]$, where C_j=completion time of job j



Example: possible schedule on 4 jobs

- Test job 1
- Schedule right away because it has large Smith ratio
- Test job 2
- Decide not to schedule yet because it has small Smith ratio
- Execute jobs 3 and 4 untested
- Execute remaining job 2

Model 1

Levi, Magnanti, Shaposhnik, Scheduling with Testing, Management Science, 2019

Notations changed for the talk

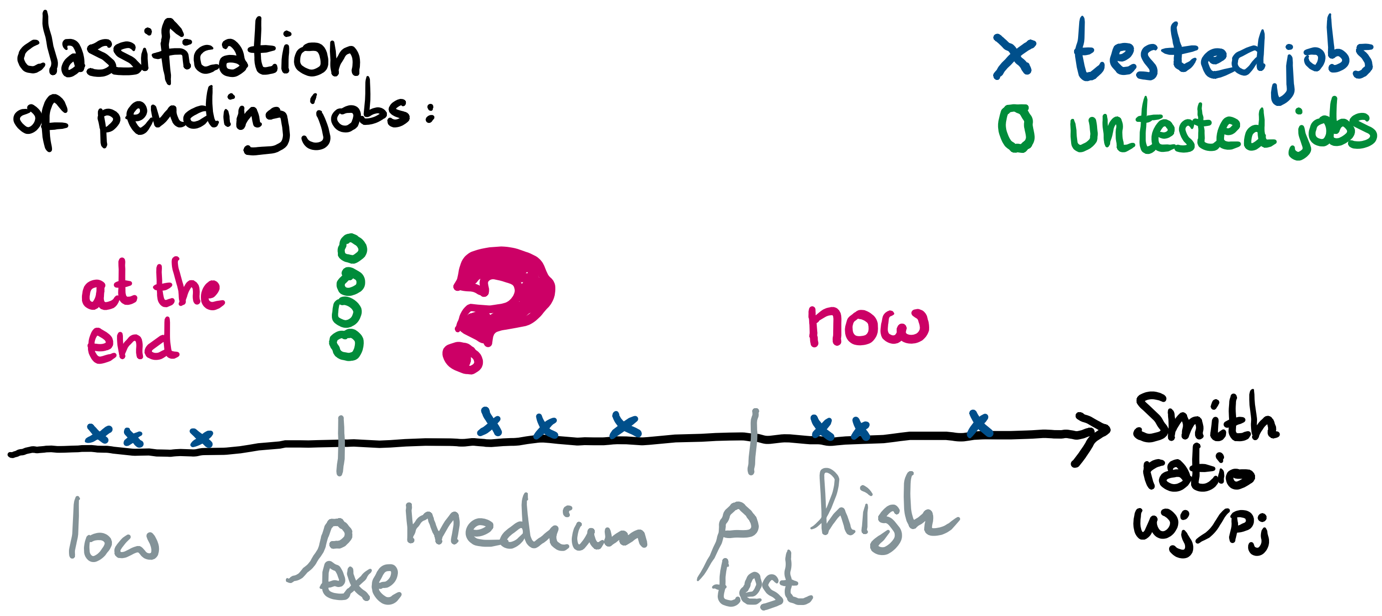
Single machine, n jobs, each job j has processing time pj, and priority weight wj.

(p_j, w_j) are randomly chosen from a joint distribution, identical for each j.

Initially only the distribution is known, not the actual job characteristics.

Algorithm can do a test for a specific job j, revealing p_j, w_j, it occupies 1 time unit on the schedule.

Objective: minimize $\mathbf{E}[\Sigma w_j C_j]$, where C_j=completion time of job j



- Expected Smith ratio $\rho_{exe} = \boldsymbol{E}\left[w_j\right]/\boldsymbol{E}\left[P_j\right]$
- Expected ratio of combined test + execution $\rho_{test} = \boldsymbol{E}\left[\mathrm{w_j}/(1+p_j)\right]$.
- It is dominant to schedule high ratio jobs right away
- At any moment algorithm needs to decide whether
  - to test a job
  - to execute a job (with highest ratio, medium ratio)

Model 1

Levi, Magnanti, Shaposhnik, Scheduling with Testing, Management Science, 2019

Notations changed for the talk

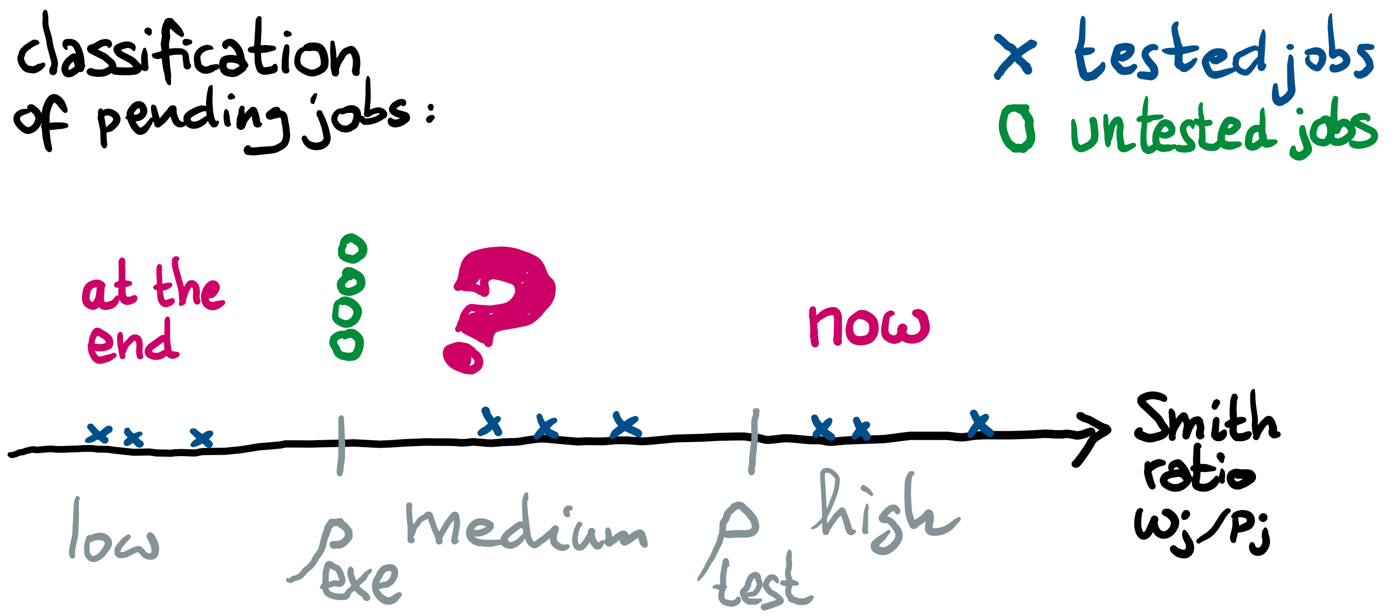
Single machine, n jobs, each job j has processing time pj, and priority weight wj.

(p_j, w_j) are randomly chosen from a joint distribution, identical for each j.

Initially only the distribution is known, not the actual job characteristics.

Algorithm can do a test for a specific job j, revealing p_j, w_j, it occupies 1 time unit on the schedule.

Objective: minimize $\mathbf{E}[\Sigma w_j C_j]$, where C_j=completion time of job j



- If $\rho_{test} \leq \rho_{exe}$ it is optimal to schedule all jobs without testing
- If $\rho_{test} > \rho_{exe}$, then it is dominant to schedule in two phases
  - Execute all high ratio jobs
  1. Test some jobs (and execute right away if they have high ratio)
  2. Execute all pending jobs (in order of decreasing Smith ratio)
- Hence algorithm only needs to decide when to switch to phase 2
- Optimal decision can be computed by dynamic programming. States contain:
  - Number of untested jobs
  - Total weight of low ratio jobs
  - Total weight of medium ratio jobs
  - Total processing time of medium ratio jobs
  - Expected cost generated by testing a job
- An FPTAS is obtained using standard rounding technique.

Motivation for an adversarial model

I know how to optimize your schedule, just give me the distribution on processing times and weights.

Hmm, we don't even have this information.

Motivation: send files,
possibly compressing them first

Model 2

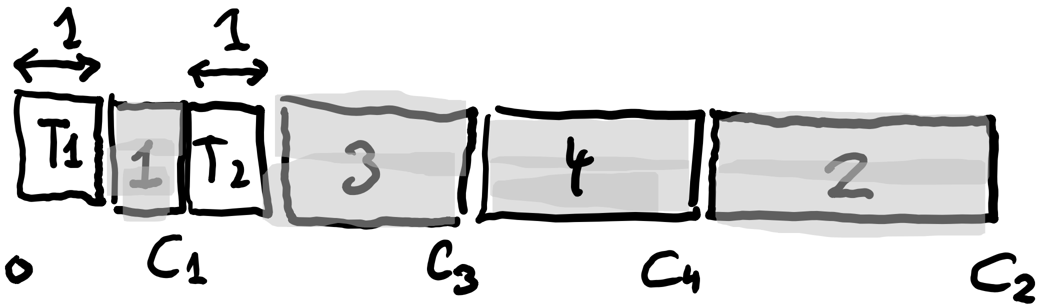
D, Erlebach, Megow, Meißner,
An adversarial model for scheduling with testing, Algorithmica, 2020.

Single machine, n jobs.

Processing time of job j = u_j if untested and p_j if tested. Only u_j is known.

A test occupies 1 unit on the schedule and reveals $p_j \in [0, u_j]$.

Cost = ΣC_j,
where C_j=completion time of job j



Example: possible schedule on 4 jobs

- Test job 1
- Schedule right away because it is short
- Test job 2
- Decide not to schedule yet because it is long
- Execute jobs 3 and 4 untested
- Execute remaining job 2

Model 2

D, Erlebach, Megow, Meißner,
An adversarial model for scheduling with testing, Algorithmica, 2020.

Single machine, n jobs.

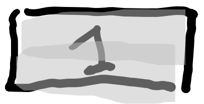
Processing time of job j = u_j if untested and p_j if tested. Only u_j is known.

A test occupies 1 unit on the schedule and reveals $p_j \in [0, u_j]$.

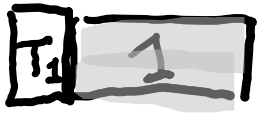
Cost = ΣC_j,
where C_j=completion time of job j

Example with a single jobs

- Schedule it untested:
  objective = u_1



- Test it, in the worst case it reveals
  $p_1 = u_1$.
  objective = $1 + \mathrm{u}_1$



Performance measure

We normalize by cost of optimal schedule

- Competitive ratio = $\max \frac{ALG(I)}{OPT(I)}$
  maximized over all instances
  $I = (p_1, \dots, p_n, u_1, \dots, u_n)$
  ALG = cost of algorithm
  OPT = cost of optimal schedule,
  i.e. test iff $1 + p_j < u_j$

- Competitive ratio
  = price of hidden information

Model 2

D, Erlebach, Megow, Meißner,
An adversarial model for scheduling with testing, Algorithmica, 2020.

Single machine, n jobs.

Processing time of job j = u_j if untested and p_j if tested. Only u_j is known.

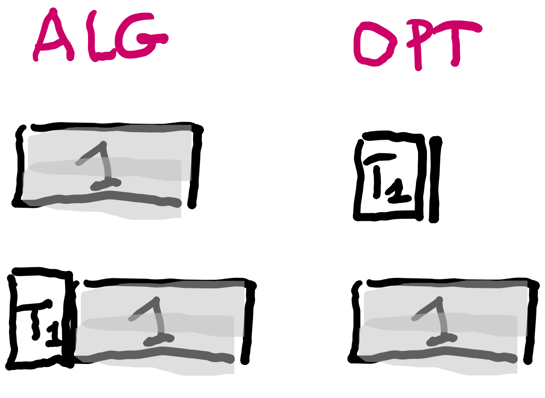
A test occupies 1 unit on the schedule and reveals $p_j \in [0, u_j]$.

Cost = ΣC_j,
where C_j=completion time of job j

Warmup with a single job

- Schedule it untested:
  in the worst case $p_j = 0$
  competitive ratio = $\frac{u_1}{1+p_j} = u_1$
- Test it, in the worst case $p_1 = u_1$.
  Competitive ratio = $\frac{1+p_1}{u_1} = 1 + \frac{1}{u_1}$

Worst case instance gives competitive ratio $\varphi = 1.618$



Model 2

D, Erlebach, Megow, Meißner,
An adversarial model for scheduling with testing, Algorithmica, 2020.

Single machine, n jobs.

Processing time of job j = u_j if untested and p_j if tested. Only u_j is known.

A test occupies 1 unit on the schedule and reveals $p_j \in [0, u_j]$.

Cost = ΣC_j,
where C_j=completion time of job j

Competitive ratio	Lower bound	Upper bound
Deterministic ratio	1.8546	2
Randomized ratio	1.6257	1.7453 (asymptotic)
Uniform $u_j = \mathrm{p}$	1.8546	1.9338
Uniform $u_j = \mathrm{p}, p_j \in \{0, p\}$	1.8546	1.8668

Model 2

D, Erlebach, Megow, Meißner,
An adversarial model for scheduling with testing, Algorithmica, 2020.

Single machine, n jobs.

Processing time of job j = u_j if untested and p_j if tested. Only u_j is known.

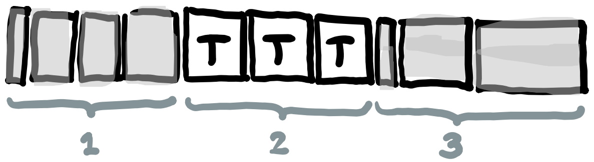
A test occupies 1 unit on the schedule and reveals $p_j \in [0, u_j]$.

Cost = ΣC_j,
where C_j=completion time of job j

Our results

Competitive ratio	Lower bound	Upper bound
Deterministic ratio	1.8546	2
Randomized ratio	1.6257	1.7453 (asymptotic)
Uniform $u_j = \mathrm{p}$	1.8546	1.9338
Uniform $u_j = \mathrm{p}, p_j \in \{0, p\}$	1.8546	1.8668

1. Execute untested all jobs j with $u_j < 2$
2. Test all other jobs.
3. Execute all tested jobs (in optimal order)



Model 2

D, Erlebach, Megow, Meißner,
An adversarial model for scheduling with testing, Algorithmica, 2020.

Single machine, n jobs.

Processing time of job j = u_j if untested and p_j if tested. Only u_j is known.

A test occupies 1 unit on the schedule and reveals $p_j \in [0, u_j]$.

Cost = ΣC_j,
where C_j=completion time of job j

Generalization for parallel identical machines
Albers, Eckl, Scheduling with Testing on Multiple Identical Parallel Machines

Our results

Competitive ratio	Lower bound	Upper bound
Deterministic ratio	1.8546	2
Randomized ratio	1.6257	1.7453 (asymptotic)
Uniform $u_j = \mathrm{p}$	1.8546	1.9338
Uniform $u_j = \mathrm{p}, p_j \in \{0, p\}$	1.8546	1.8668

Game played between adversary and algorithm. Uniform instance ($u_j = \mathrm{p}$)

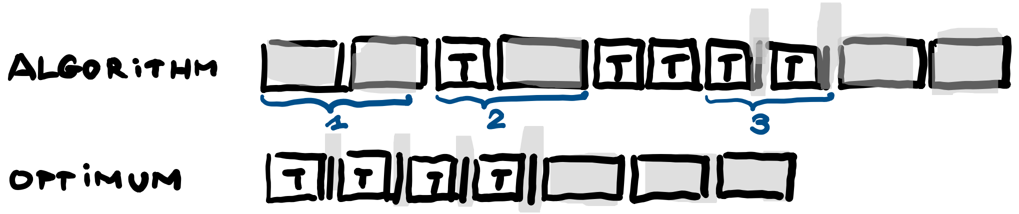
Algorithm decides:

1. How many jobs to execute untested (adversary makes them short, i.e., $p_j = 0$).
2. Among the tested long jobs ($p_j = p$) how many will be executed right after their test

Adversary decides:

3. How many tested jobs are short

Second order analysis of minimizer/maximizer of the competitive ratio



Motivation: dry run jobs on far high-speed server to learn processing time

Model 3

D, Dufossé, Nadal, Trystram, Vásquez. Scheduling with a processing time oracle, submitted, 2019

Single machine, n jobs, each job j has processing time p or p+x.

Initially only n, p, x are known, not the individual processing time.

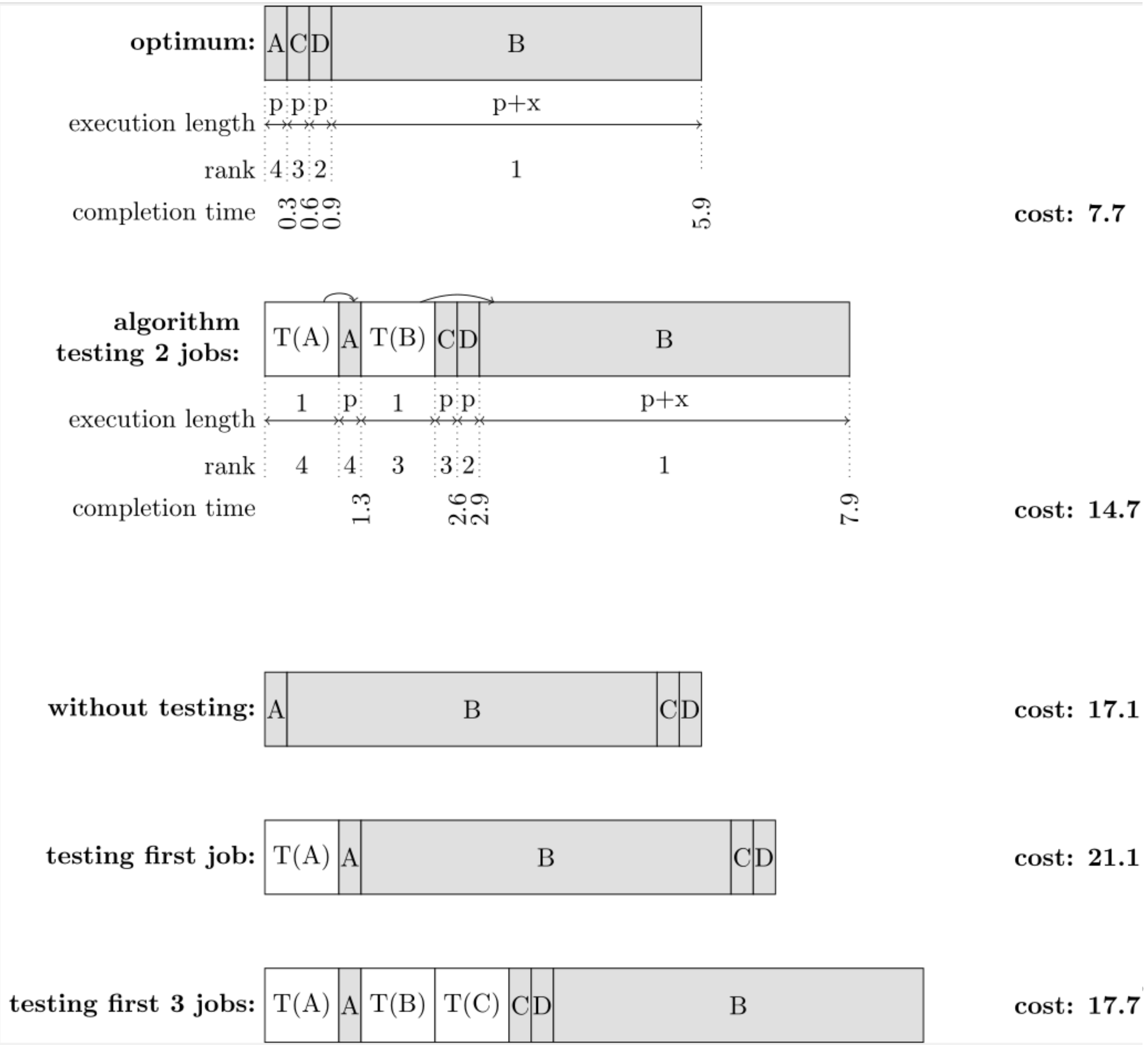
Algorithm can do a test for a specific job j, revealing if it is short or long, it occupies 1 time unit on the schedule.

Objective: minimize ΣC_j,
where C_j=completion time of job j

Goal: minimize competitive ratio
CR:=cost of schedule produced by algorithm over cost of optimal schedule

Example

- 4 jobs A,B,C,D, only B is long (but algorithm does not know this initially)
- p=0.3, x=4.7



Model 3

D, Dufossé, Nadal, Trystram, Vásquez. Scheduling with a processing time oracle, submitted, 2019

Single machine, n jobs, each job j has processing time p or p+x.

Initially only n, p, x are known, not the individual processing time.

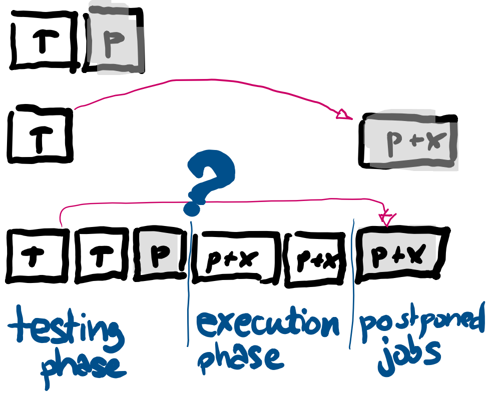
Algorithm can do a test for a specific job j, revealing if it is short or long, it occupies 1 time unit on the schedule.

Objective: minimize ΣC_j,
where C_j=completion time of job j

Goal: minimize competitive ratio
CR:=cost of schedule produced by algorithm over cost of optimal schedule

What we know

- It is dominant to execute tested short jobs immediately after their test.
- It is dominant to postpone the execution of tested long jobs towards the end.
- Algorithm only needs to decide for every job: test or execute untested



What we conjecture

- Two phases: optimal algorithm tests some jobs, then executes untested all remaining jobs

Warmup: non adaptive algorithm

- Algorithm decides before hand how many jobs to test
- Adversary (generating worst case instance) decides how many untested jobs are short and how many tested jobs are short
- Second order analysis -> optimal algorithm (assuming the conjecture)

Model 3

D, Dufossé, Nadal, Trystram, Vásquez. Scheduling with a processing time oracle, submitted, 2019

Single machine, n jobs, each job j has processing time p or p+x.

Initially only n, p, x are known, not the individual processing time.

Algorithm can do a test for a specific job j, revealing if it is short or long, it occupies 1 time unit on the schedule.

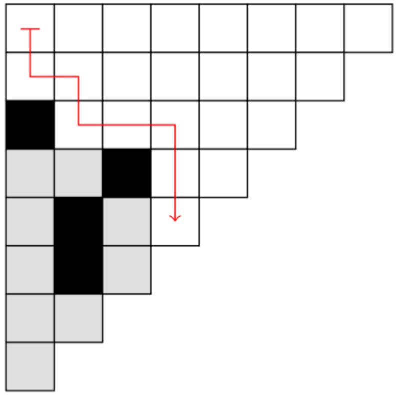
Objective: minimize ΣC_j,
where C_j=completion time of job j

Goal: minimize competitive ratio
CR:=cost of schedule produced by algorithm over cost of optimal schedule

Adaptive algorithm

- Assuming conjecture
- Algorithm=decide when to stop testing jobs
- Adversary=decide for each tested job, whether it is short or long

Algorithm-Adversary interaction modeled as a path



- Start in upper left cell
- Tested short job = one step down
  Tested long job = one step right
- To each cell along the path,
  we associate a stop ratio
  = competitive ratio obtained
  if algorithm stops here
- Algorithm will stop at cell with minimal stop ratio
- Suppose adversary know a strategy which forces ratio R*
- Mark cells (black) which adversary should avoid to force ratio > R*
- Marked cells form a combinatorial tableau, its boundary is the next path the adversary tries
- Leads to an $O(n^3)$ algorithm to compute optimal strategies for both algorithm and adversary

Research directions

Allow machine learning based tests, which are prone to errors.

Implement in job scheduler of a cluster, and measure effect of job length predictions.

These model make sense only if tests are long compared to job processing times.