An Adversarial Model for Scheduling with Testing

Optimizing with explorable uncertainty

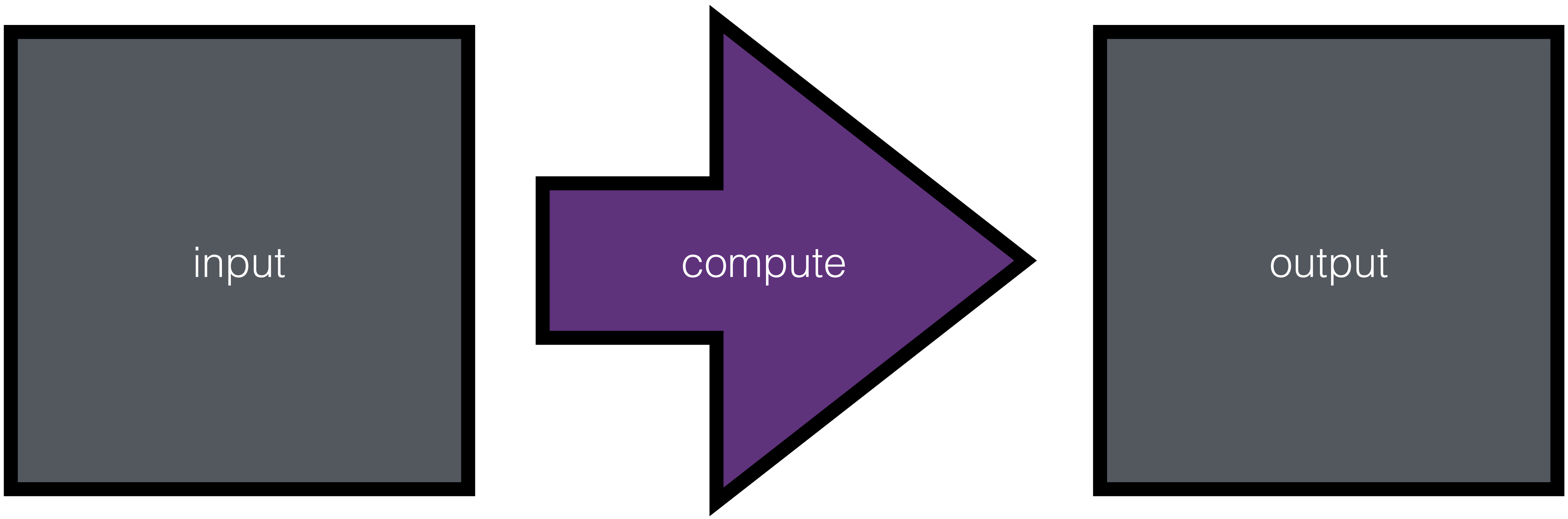
C. Dürr
University Pierre et Marie Curie, Paris-6

LAGOS 2017

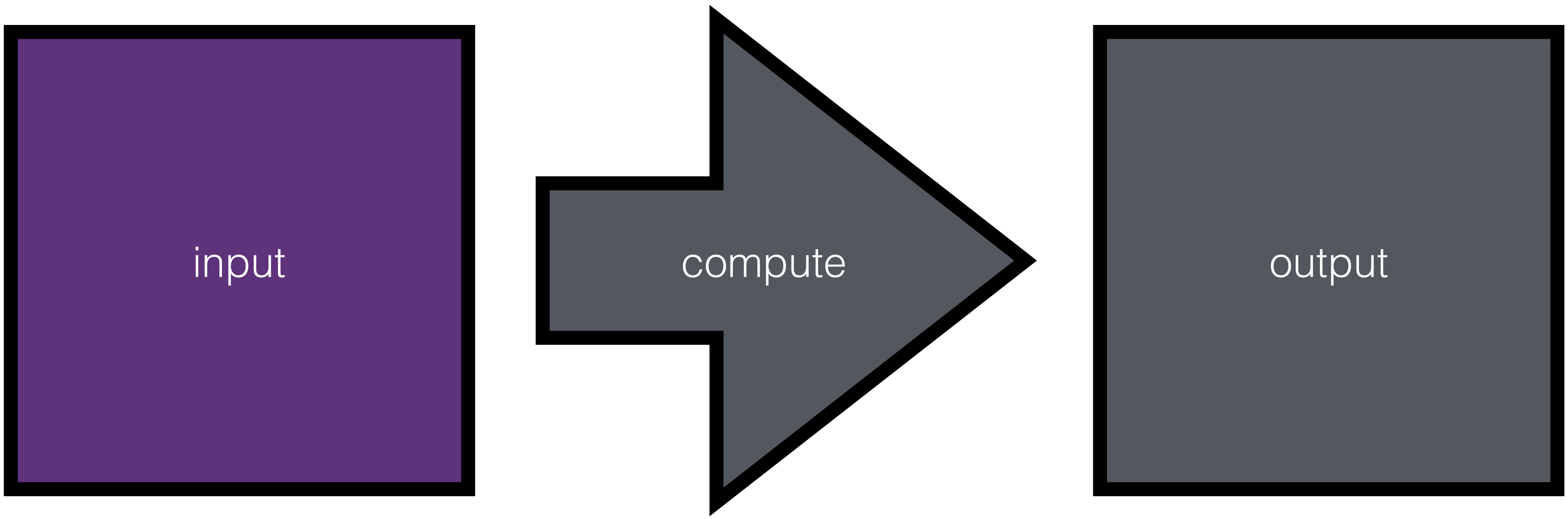
Outline

1. The model
2. Minimum Spanning Tree
3. Scheduling

Computing paradigm

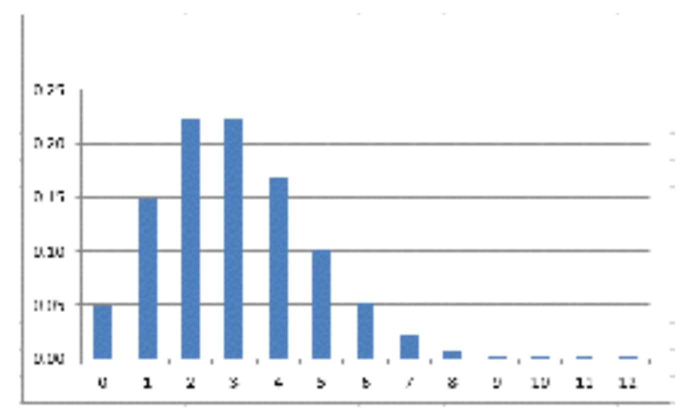


Computing paradigm



Models with uncertainty

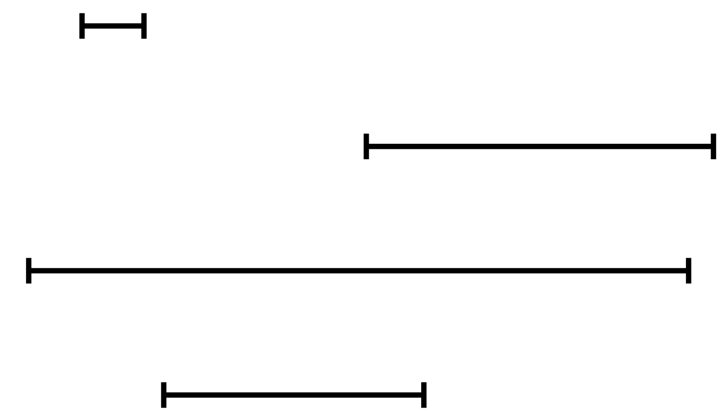
- Stochastic optimization
- Robust optimization
- Some papers with queries
  - A model for data in motion,
    Simon Kahan, STOC'1991
  - The Trapp system,
    Olston and Widom, VLDB'2008
  - Minimum spanning Trees,
    Erlebach et al, STACS'2008



- Input is drawn from known distribution
- Need to produce a solution minimizing expected objective value

Models with uncertainty

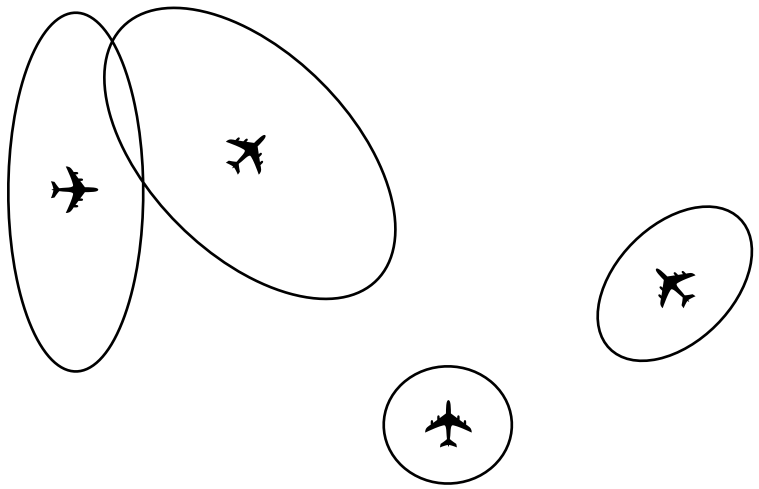
- Stochastic optimization
- Robust optimization
- Some papers with queries
    - A model for data in motion, Simon Kahan, STOC’1991
    - The Trapp system, Olston and Widom, VLDB’2008
    - Minimum spanning Trees, Erlebach et al, STACS'2008



- Input is drawn from known set (scenarios)
- Need to produce a solution minimizing the worst objective value over all scenarios

Models with uncertainty

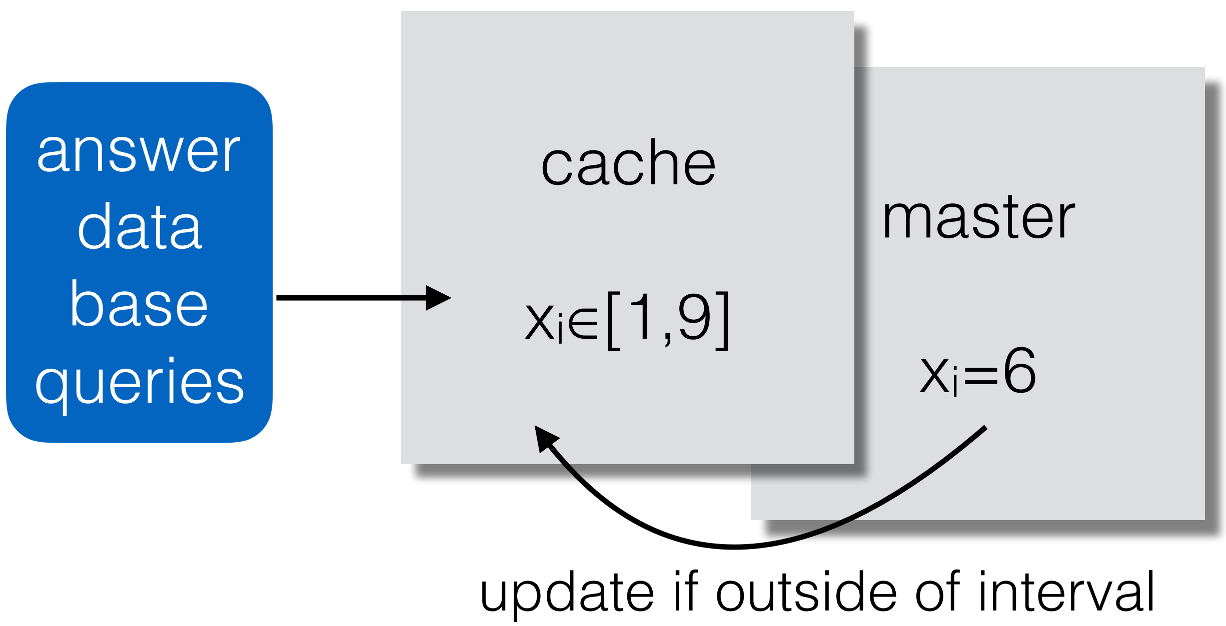
- Stochastic optimization
- Robust optimization
- Some papers with queries
  - A model for data in motion, Simon Kahan, STOC'1991
  - The Trapp system, Olston and Widom, VLDB'2008
  - Minimum spanning Trees, Erlebach et al, STACS'2008



- Points are moving in space
- Each point lays in a set, determined by last known position and velocity
- Query minimal number of point positions in order to solve some problem
- see also Bruce et al, ToCS'2005

Models with uncertainty

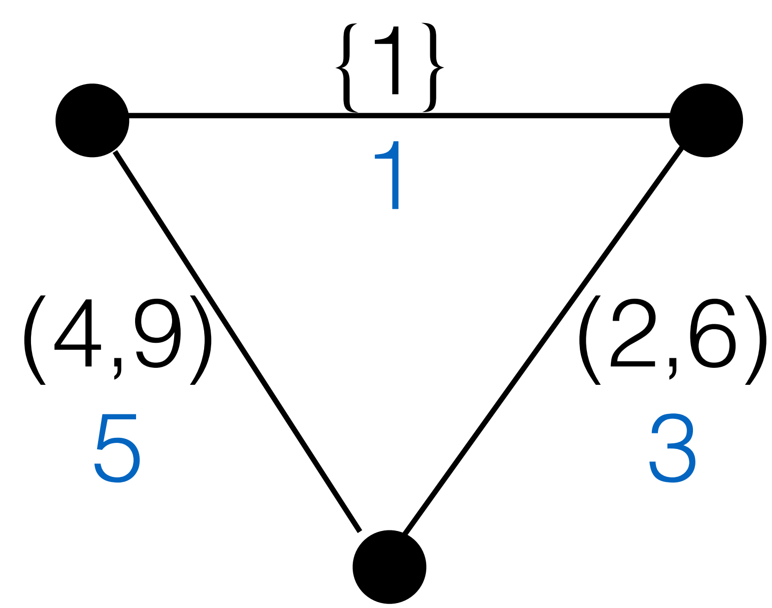
- Stochastic optimization
- Robust optimization
- Some papers with queries
  - A model for data in motion, Simon Kahan, STOC'1991
  - The Trapp system, Olston and Widom, VLDB'2008
  - Minimum spanning Trees, Erlebach et al, STACS'2008



- local cache contains intervals of values
- master server contains exact values
- data base works with intervals, only querying the master server when more precision is required

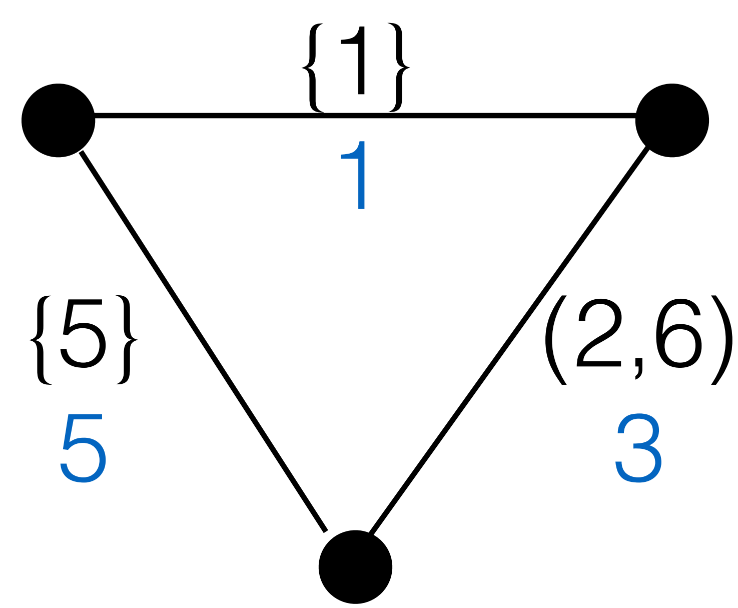
Minimum spanning tree

- **given**: graph,
  open edge weight intervals
- **hidden**: exact edge weights
- **query**: reveals exact edge weight
- **goal**: identify a minimum spanning tree with minimal number of queries



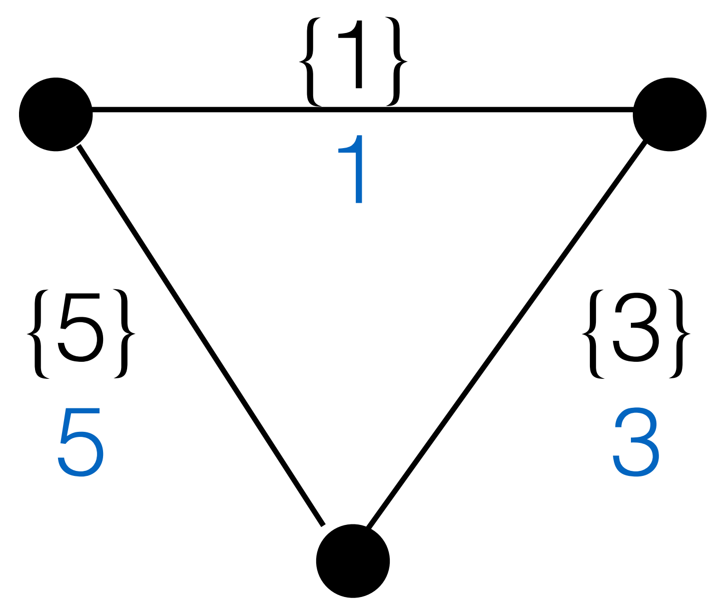
Minimum spanning tree

- **given**: graph,
  open edge weight intervals
- **hidden**: exact edge weights
- **query**: reveals exact edge weight
- **goal**: identify a minimum spanning tree with minimal number of queries



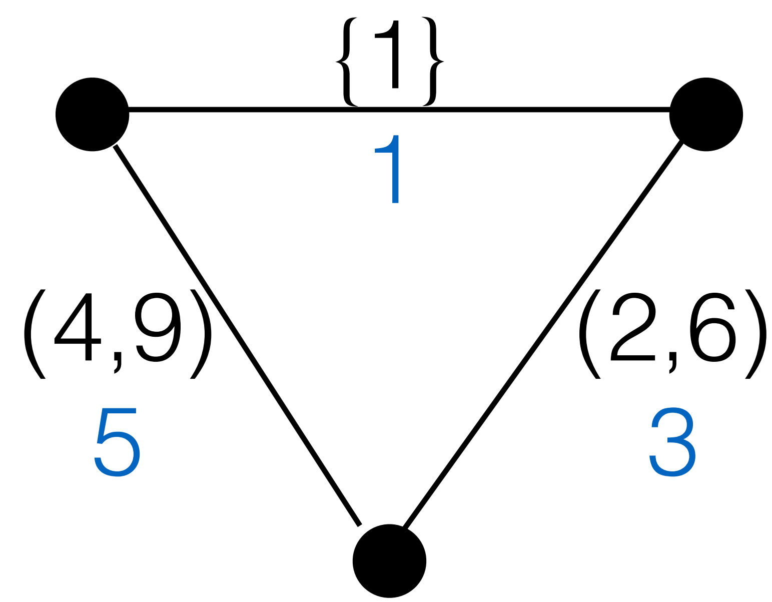
Minimum spanning tree

- **given**: graph,
  open edge weight intervals
- **hidden**: exact edge weights
- **query**: reveals exact edge weight
- **goal**: identify a minimum spanning tree with minimal number of queries



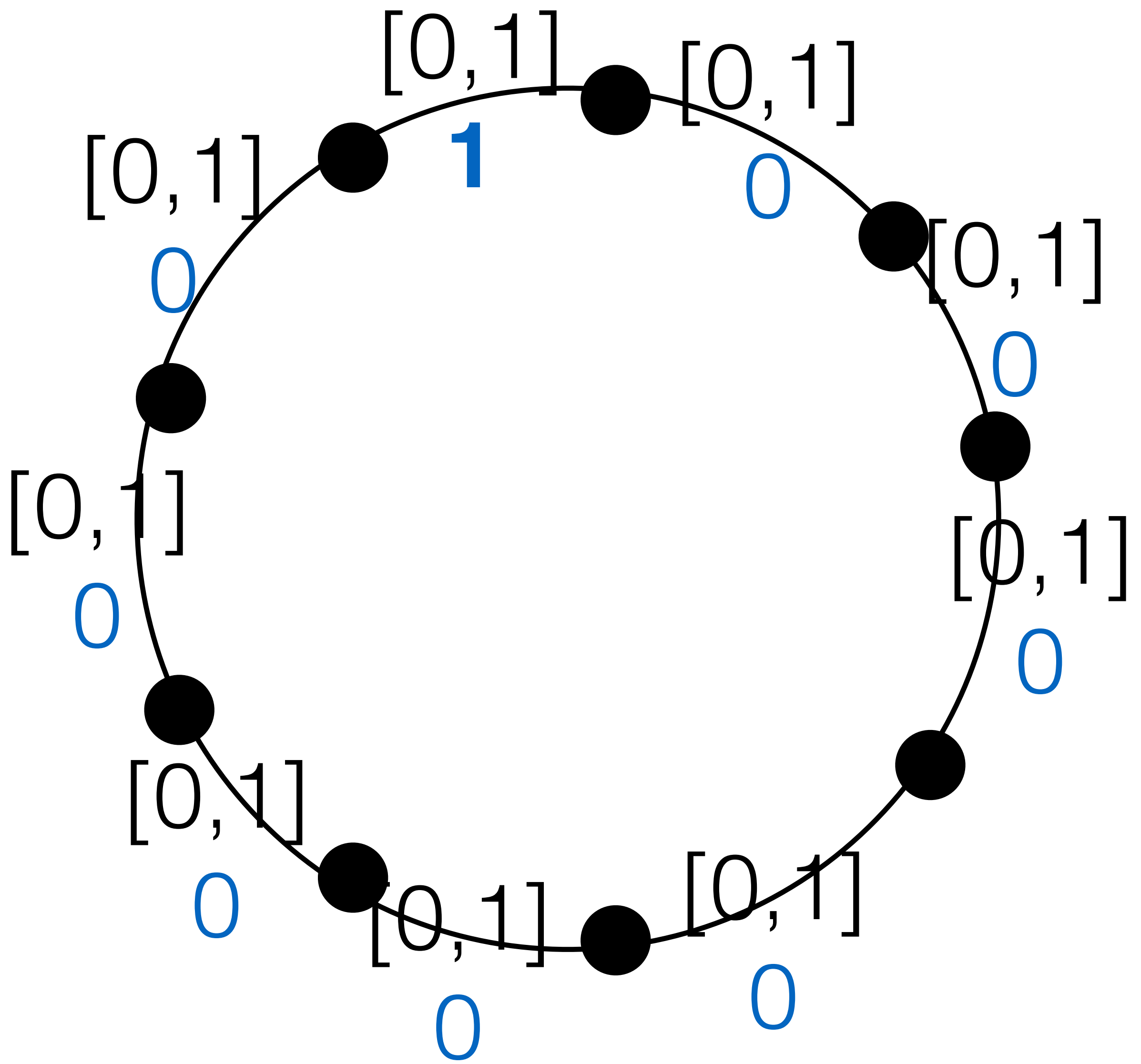
Minimum spanning tree

- **measure**: An algorithm ALG is c-competitive if for all instances I
  $ALG_I \leq c\ OPT_I$
- For *asymptotic* competitive ratio an additive constant is allowed
- **OPT_I**: minimal number of queries, say an adversary could make if he knew the exact values but still need to query them



Why *open* intervals?

- **if uncertainty intervals were closed**: Consider this graph.
- **OPT$_\mathbf{I}$**: is 1
- **ALG**: is n-1
- **Ratio**: terrible large



Minimum spanning tree

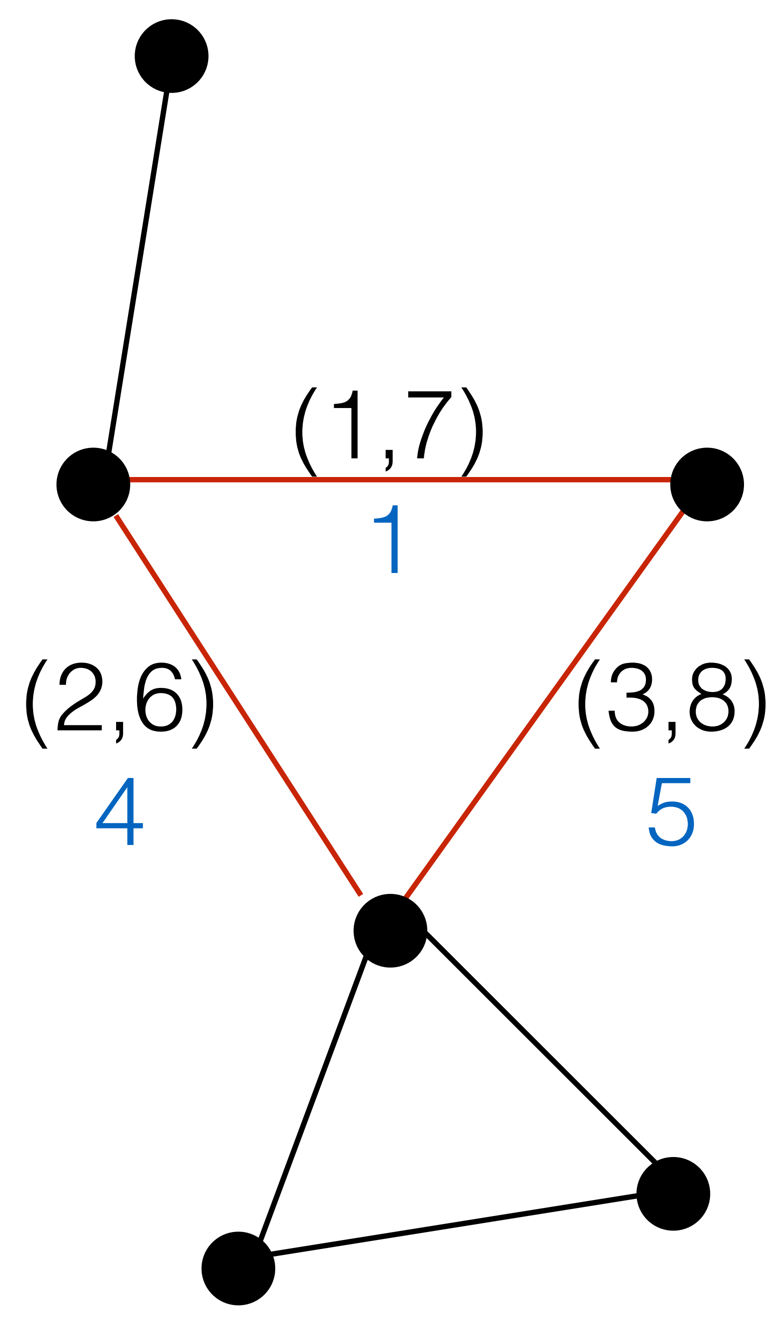
competitive ratio	lower bound	upper bound
deterministic [Erlebach et al. STACS'2008]	2	2
randomized [Megow,Meißner,Skutella, ESA'2015]	1,5	1,707

Minimum spanning tree

competitive ratio	lower bound	upper bound
deterministic [Erlebach et al. STACS'2008]	2	2
randomized [Megow,Meißner,Skutella, ESA'2015]	1,5	1,707

The witness algorithm

- For a more general setting: **Cheapest Set Problem**
- Find a *feasible* set $S \subseteq \{1,..,n\}$ minimizing $\Sigma_{i \in S}\, x_i$
- W is a **witness set** if it impossible to solve the problem without querying at least one element from W.
- *Algorithm*: **While** instance not solved: **choose** a witness set and **query** all items from it.

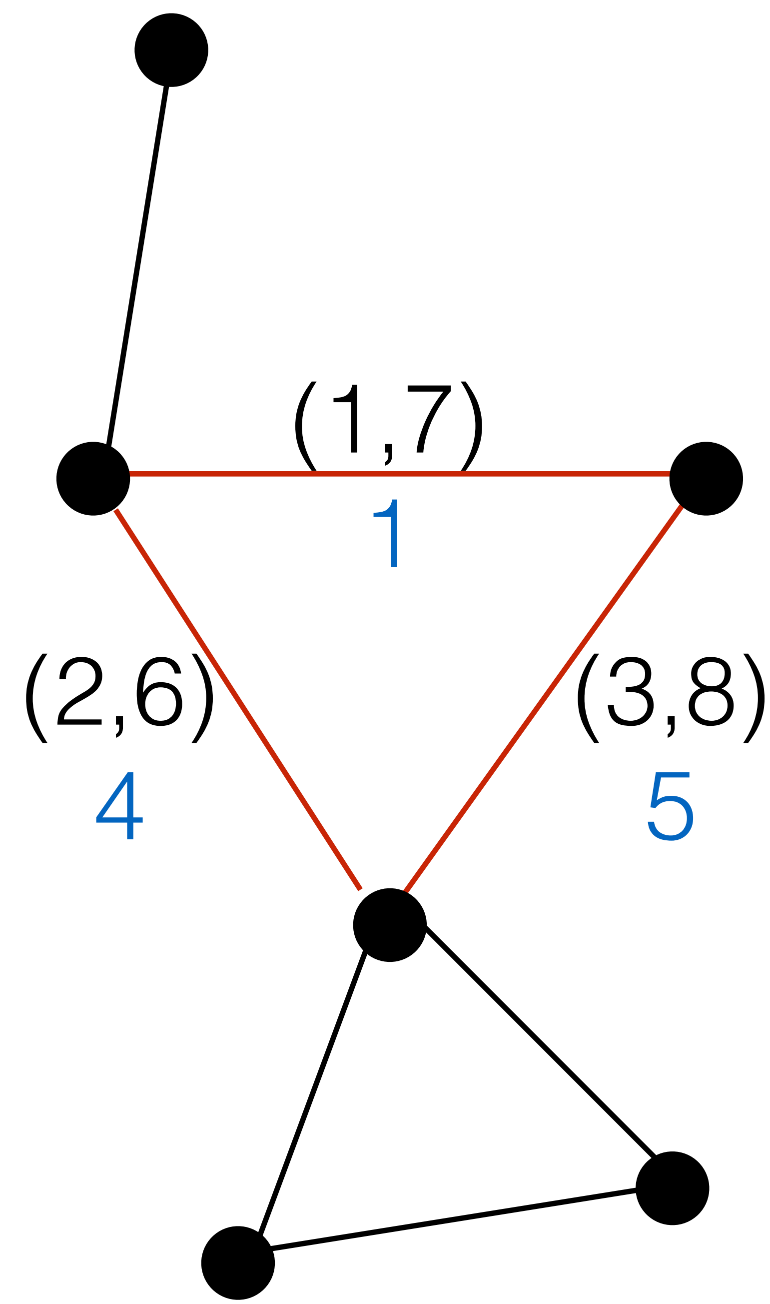


The witness algorithm

- Lemma If each chosen witness set has size ≤ c, then the algorithm is c-competitive

- W is a **witness set** if it impossible to solve the problem without querying at least one element from W.

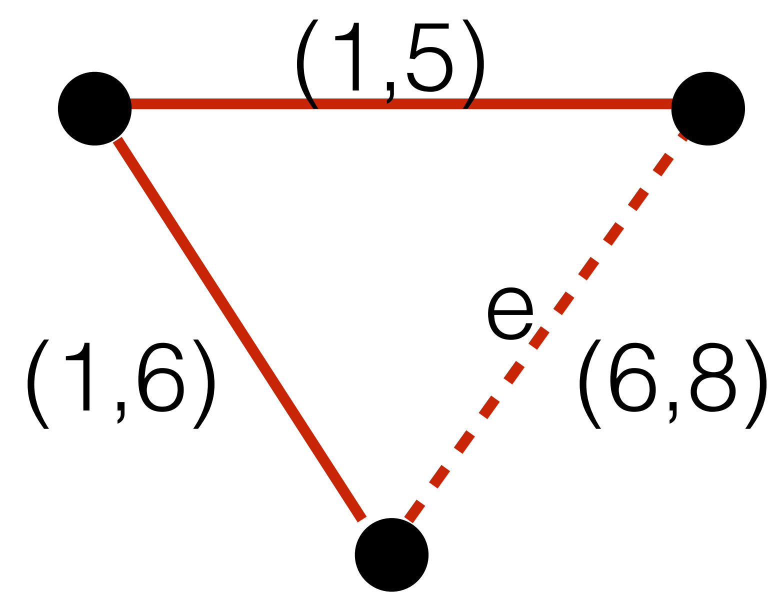
- *Algorithm*: **While** instance not solved: **choose** a witness set and **query** all items from it.



The U-Red algorithm

- given: for edges e $w_e \in (L_e, U_e)$

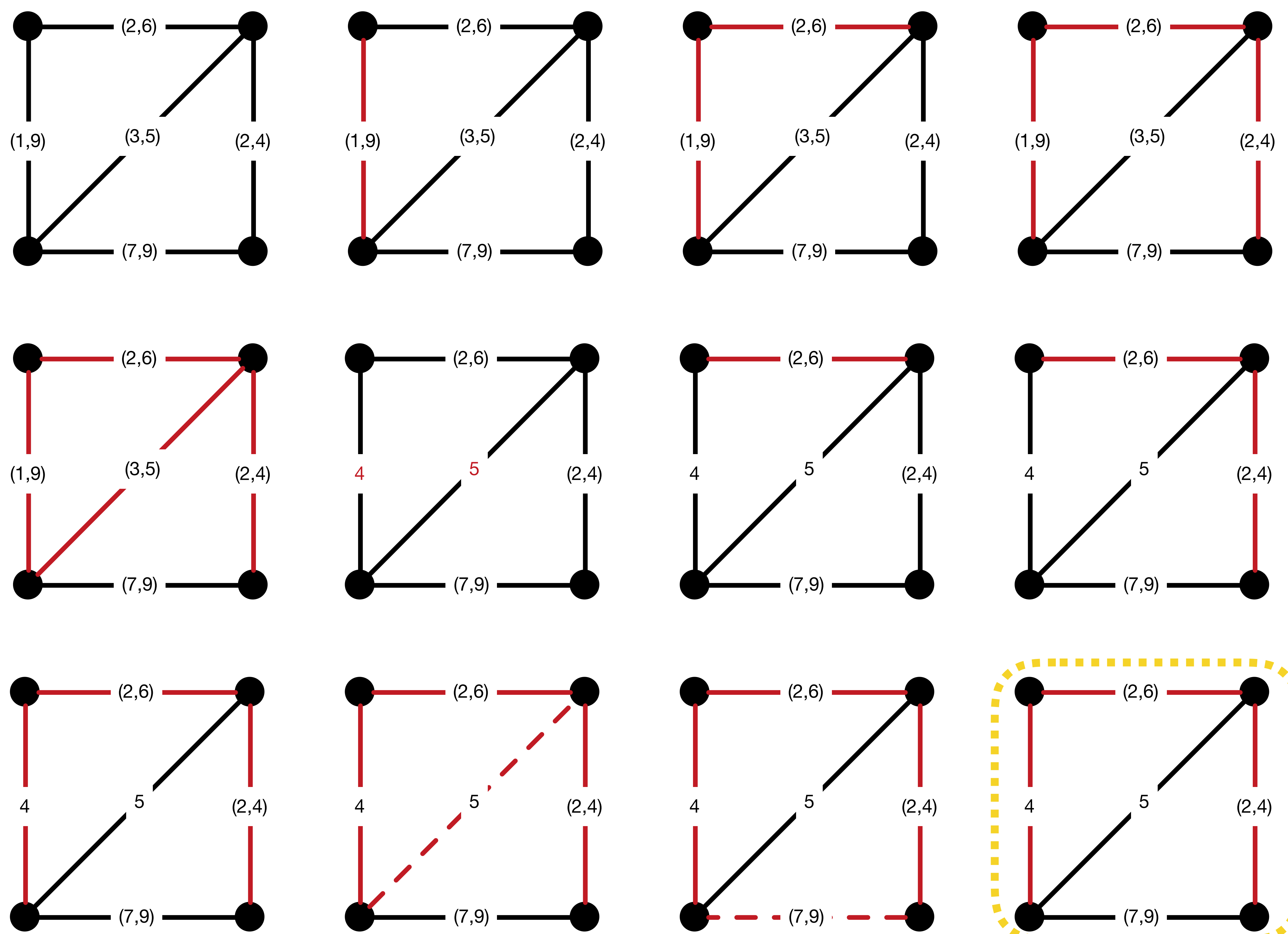
- Red rule: if there is an edge e in a cycle C with $L_e \geq U_f$ for all $f \in C \setminus e$ (*always maximal edge*), then there is a minimum spanning tree without e



- **U-Red**: initially T=empty
  for all edges e in lexicographically increasing (L_e, U_e) order:
    add e to T
    if T has a cycle C
      if e is always maximal in C
        remove from T
      else
        let $f \in C$ s.t. U_f is maximal
        let $g \in C \setminus f$ s.t. $U_g > L_f$
        query f and g, and restart
  return T

Key argument: {f,g} is a **witness set**, hence the algorithm is 2-competitive

The U-Red algorithm



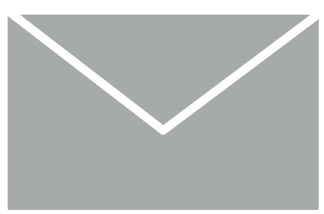
Some personal work

- So far: minimize query cost to compute optimal solution
- Now: add query cost to objective value
- →find compromise between querying and improving solution
- joint work with Thomas Erlebach, Nicole Megow and Nicole Meißner

Warmup

has to send a
single file

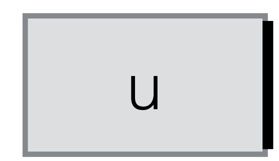
cares about the
reception time



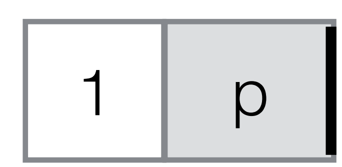
could compress it
before sending

This is a scheduling problem

- Single job, has upper limit u
- **Either** schedule untested : cost u



- **or** test (takes 1 unit), which reveals processing time $0 \le p \le u$, and schedule it : cost 1+p



u is given
p is hidden
a test reveals p

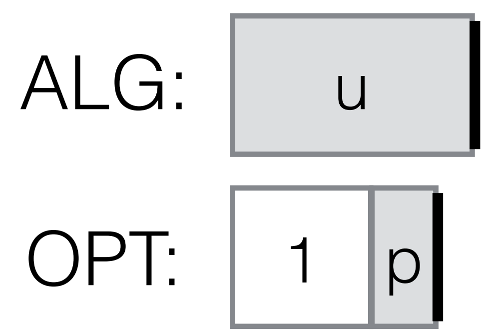
Minimize competitive ratio

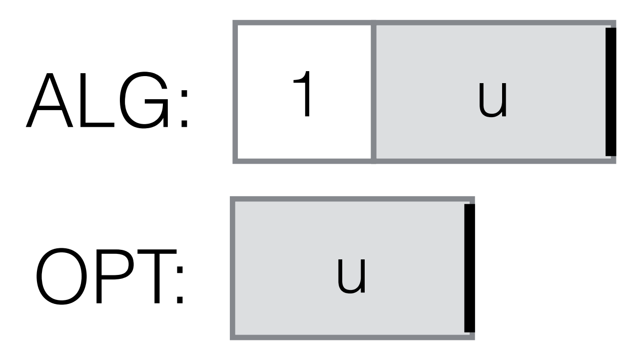
compare algorithm with an adversary who knows p, and therefore knows if it is worth to test

- Produce a solution with a guaranty on the cost compared to the optimal solution

- The adversary computes an optimal solution. He knows p, but still needs to test the job, if he wants to schedule it at length p.

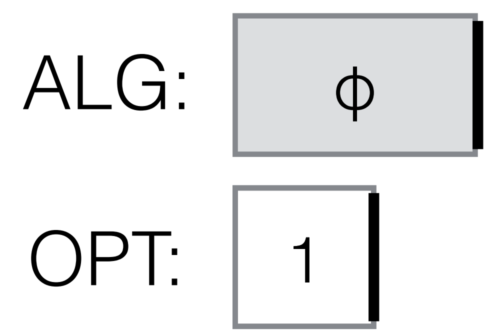
- Ratio ALG/OPT over worst instance =**competitive ratio** =price of not knowing p



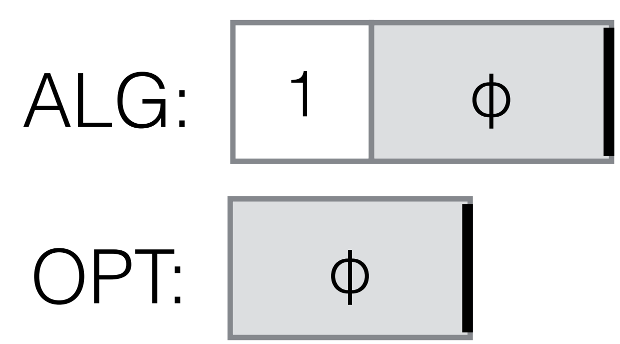


Minimize competitive ratio

- Adversary chooses u=ϕ=golden ratio=1.618… (satisfies $\phi+1=\phi^2$)



- **If** algorithm does not test, adversary chooses p=0 and tests



- **If** algorithm does test, adversary chooses p=ϕ and does not test

The general problem

has to send files
of various sizes

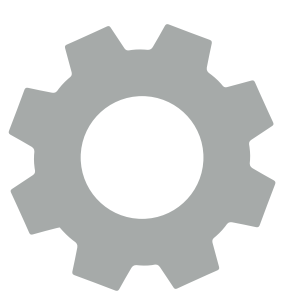
cares about ΣC_j
C_j = reception
time of file j

could compress
files before
sending

Other motivations

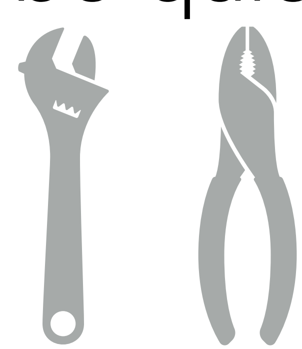
Code optimizer

machine could run a code optimizer before executing a program



safe problem resolution versus heuristic

There are two methods to solve a problem. A safe one and a heuristic that might be quicker or fail

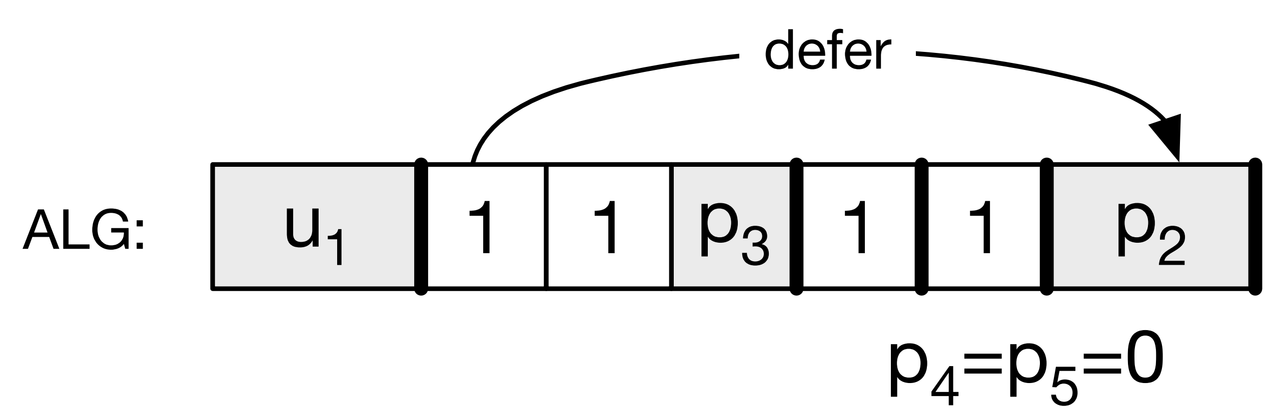


Scheduling medical appointements

quick diagnosis can estimate processing times

The general problem

- **Input**: n jobs with upper limits $u_1,\ldots,u_n$
- **Produce** a schedule consisting of job executions or tests. Test of job j takes 1 time unit and changes its processing time to $0 \le p_j \le u_j$. Can be scheduled anytime after its test.



- Objective = total completion time of jobs.
- **Minimize** ratio Objective / optimal objective
- Notice: if the goal were to minimize objective, one would never test

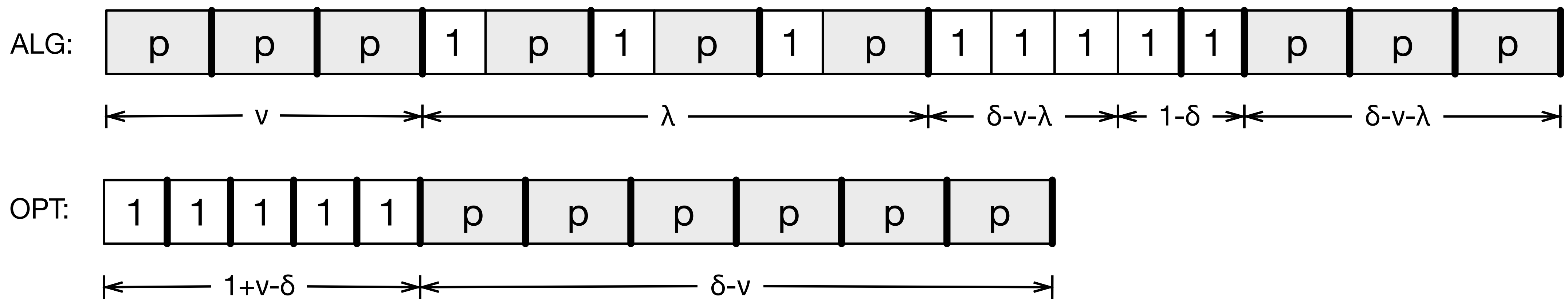
Our results

competitive ratio	lower bound	upper bound	algorithm
deterministic ratio	1.8546	2	THRESHOLD
randomized ratio	1.6257	1.7453 (asymptotic ratio)	RANDOM
det. ratio. on uniform instances (u_j=p)	1.8546	1.9338	BEAT
det. ratio. on extreme uniform instances (u_j=p, $p_j \in \{0,p\}$)	1.8546	1.8668	UTE
det. ratio on extreme uniform instances with u=1.9896	1.8546	1.8552	UTE

Our results

competitive ratio	lower bound	upper bound	algorithm
deterministic ratio	1.8546	2	THRESHOLD
randomized ratio	1.6257	1.7453 (asymptotic ratio)	RANDOM
det. ratio. on uniform instances (u_j=p)	1.8546	1.9338	BEAT
det. ratio. on extreme uniform instances (u_j=p, $p_j \in$ {0,p})	1.8546	1.8668	UTE
det. ratio on extreme uniform instances with u=1.9896	1.8546	1.8552	UTE

Deterministic lower bound



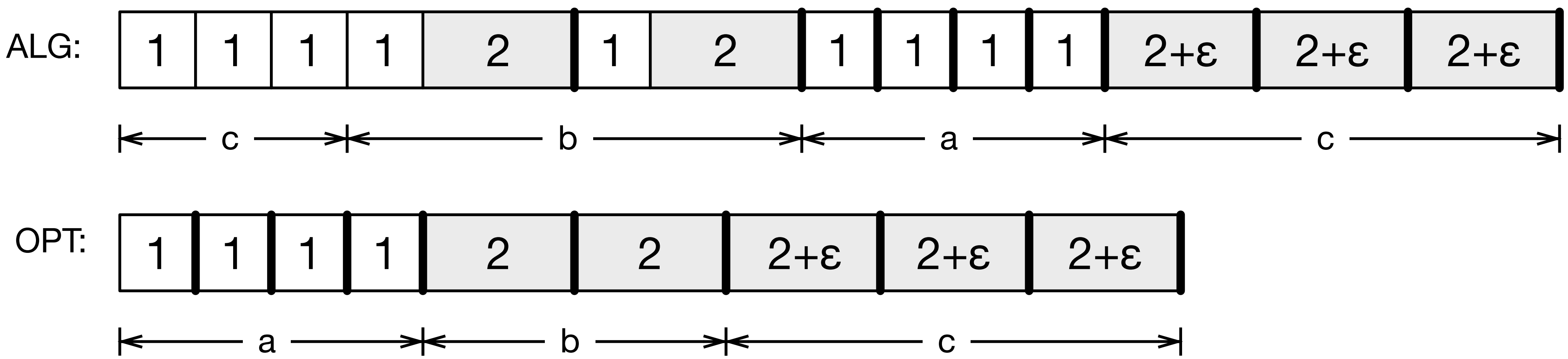
- n uniform jobs with upper limit p
- Index jobs in order they are touched by algorithm (tested or executed untested)
- p_j=0 if j≥δn or job j is executed untested by algo.
  p_j=p otherwise
- Algorithm gets even to know δ

- Any decent algorithm produces a schedule with above structure for parameters v, λ with v+λ≤δ
- The competitive ratio is
  ALG(δ,v, λ,n) / OPT(δ,v,n)
- Algorithm (minimizer) chooses v,λ
- Adversary (maximizer) chooses n,δ
- Analyzing local optima yields ratio 1.854628

Our results

competitive ratio	lower bound	upper bound	algorithm
deterministic ratio	1.8546	2	THRESHOLD
randomized ratio	1.6257	1.7453 (asymptotic ratio)	RANDOM
det. ratio. on uniform instances (u_j=p)	1.8546	1.9338	BEAT
det. ratio. on extreme uniform instances (u_j=p, $p_j \in$ {0,p})	1.8546	1.8668	UTE
det. ratio on extreme uniform instances with u=1.9896	1.8546	1.8552	UTE

Algorithm THRESHOLD



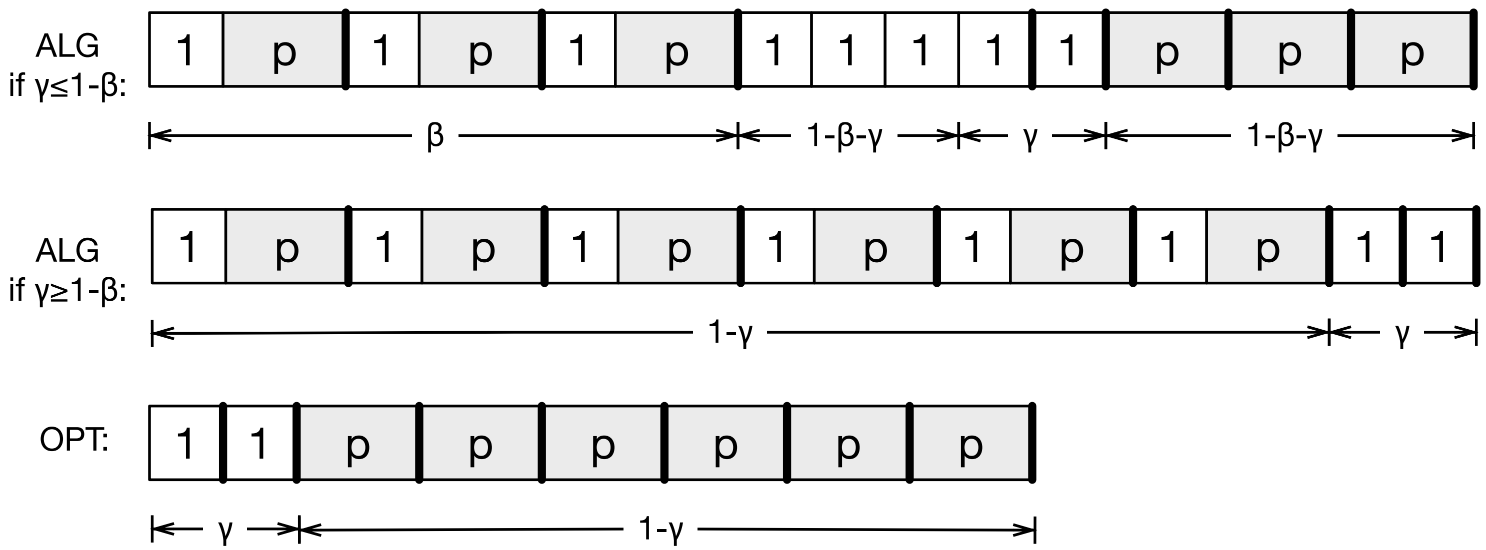
- Execute untested all jobs j with $u_j<2$ in order…
- Test all other jobs in arbitrary order. If $p_j \leq 2$, execute, otherwise defer.
- Execute all deferred jobs in order…

- **Worst case** instance:
  a jobs $u_j=2, p_j=0$
  b jobs $u_j=p_j=2$
  c jobs $u_j=p_j=2+\varepsilon$
- Simple arithmetics:
  $ALG(a,b,c) \leq 2 \cdot OPT(a,b,c)$

Our results

competitive ratio	lower bound	upper bound	algorithm
deterministic ratio	1.8546	2	THRESHOLD
randomized ratio	1.6257	1.7453 (asymptotic ratio)	RANDOM
det. ratio. on uniform instances (u_j=p)	1.8546	1.9338	BEAT
det. ratio. on extreme uniform instances (u_j=p, $p_j \in \{0,p\}$)	1.8546	1.8668	UTE
det. ratio on extreme uniform instances with u=1.9896	1.8546	1.8552	UTE

Algorithm UTE



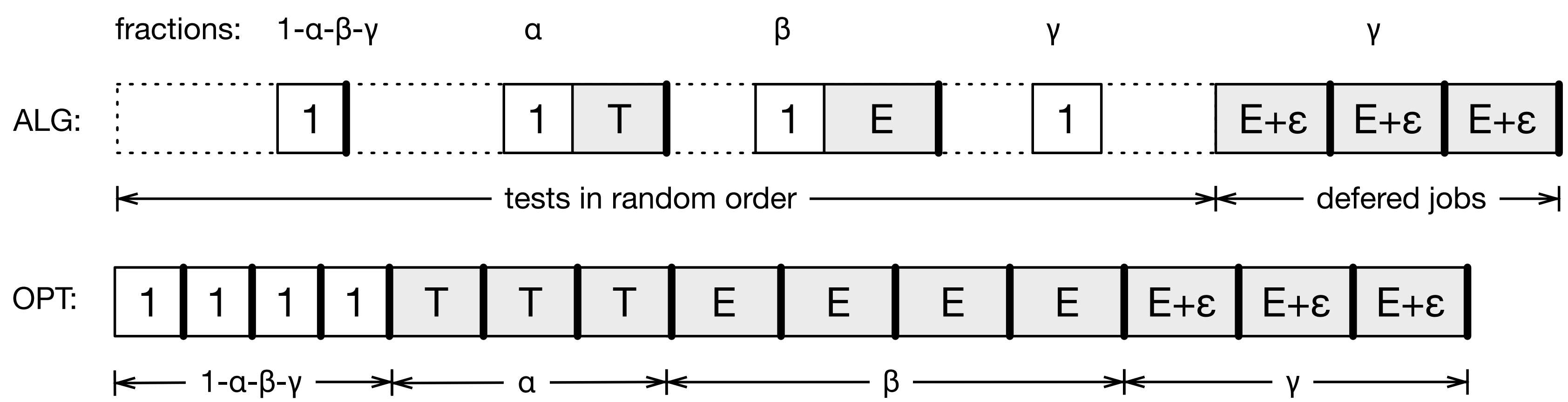
- for extrem uniform instances, $u_j=p$, $p_j \in \{0,p\}$
- has ratio $\rho = \frac{1+\sqrt{3+2\sqrt{5}}}{2} \approx 1.8668.$
- Parameter $\beta = \dfrac{1-\bar{p}+\bar{p}^2-\rho+2\bar{p}\rho-\bar{p}^2\rho}{1-\bar{p}+\bar{p}^2-\rho+\bar{p}\rho}$
- Execute all jobs untested if $p \le \rho$
- Otherwise test all jobs. Execute right after their test the first max{0,β} fraction of jobs. Then only if $p_j=0$. Finally execute deferred jobs.

- **Worst case** instance defined by
  length p
  fraction γ: the first γn tested jobs have $p_j=p$ and the remaining $p_j=0$
- Second order analysis to optimize p,γ and β

Our results

competitive ratio	lower bound	upper bound	algorithm
deterministic ratio	1.8546	2	THRESHOLD
randomized ratio	1.6257	1.7453 (asymptotic ratio)	RANDOM
det. ratio. on uniform instances (u_j=p)	1.8546	1.9338	BEAT
det. ratio. on extreme uniform instances (u_j=p, $p_j \in \{0,p\}$)	1.8546	1.8668	UTE
det. ratio on extreme uniform instances with u=1.9896	1.8546	1.8552	UTE

Algorithm RANDOM



- Algorithm **RANDOM**: Parameters $T \geq E$
  Schedule untested all jobs with upper limit $< T$ in increasing upper limit order
  Test in random order all larger jobs j, if $p_j \leq E$ execute immediately, else defer their execution
  Finally schedule deferred jobs
  in increasing processing time order

- Worst case instances:
  (1-α-β-γ) fraction of jobs : $u_j=T$, $p_j=0$
  αn jobs have $u_j=T$, $p_j=T$
  βn jobs have $u_j=E$, $p_j=E$
  γn jobs have $u_j=E+\varepsilon$, $p_j=E+\varepsilon$

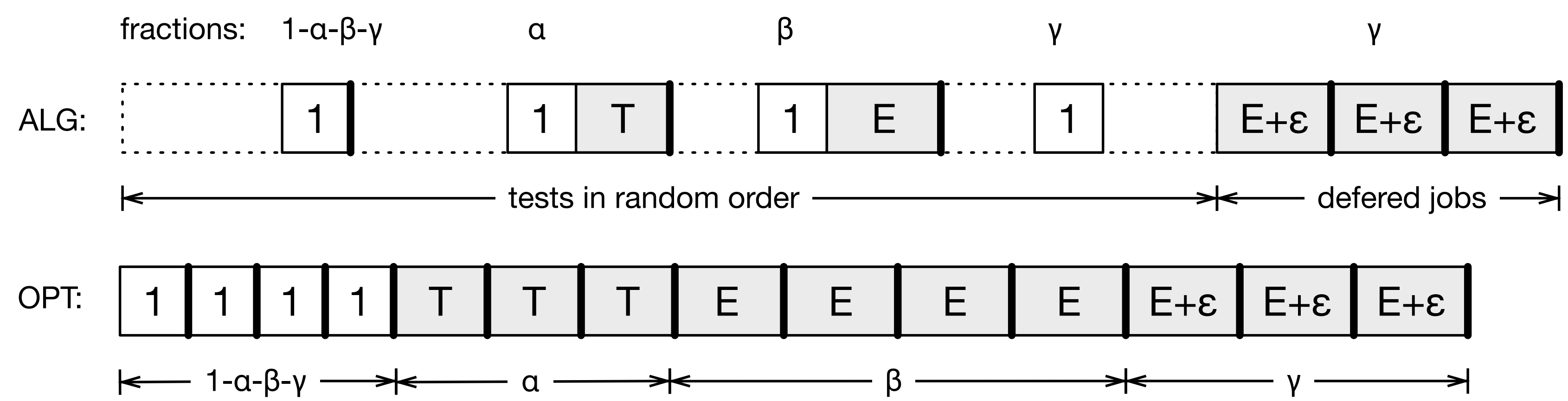
- Ratio $\leq$ T iff
  $G := OPT \cdot T - ALG \geq 0$

- Algorithm chooses T, E to maximize G
  Adversary chooses α,β,γ to minimize G

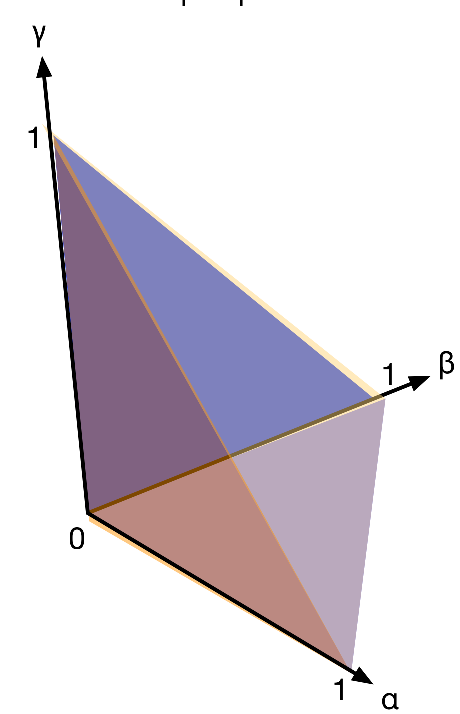
Our results

competitive ratio	lower bound	upper bound	algorithm
deterministic ratio	1.8546	2	THRESHOLD
randomized ratio	1.6257	1.7453 (asymptotic ratio)	RANDOM
det. ratio. on uniform instances (u_j=p)	1.8546	1.9338	BEAT
det. ratio. on extreme uniform instances (u_j=p, $p_j \in \{0,p\}$)	1.8546	1.8668	UTE
det. ratio on extreme uniform instances with u=1.9896	1.8546	1.8552	UTE

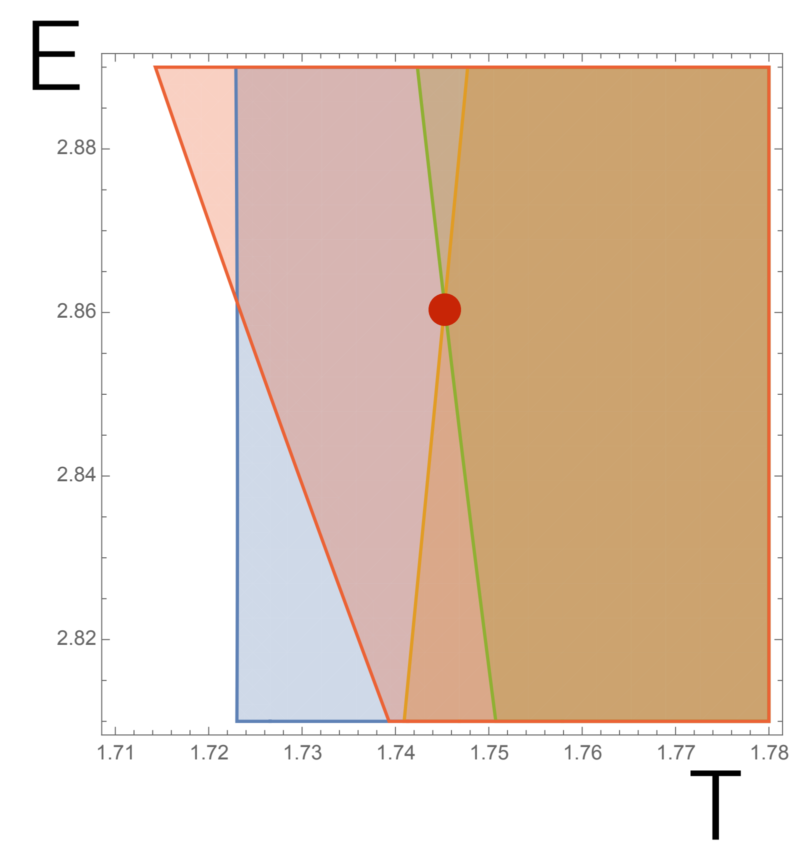
Algorithm RANDOM



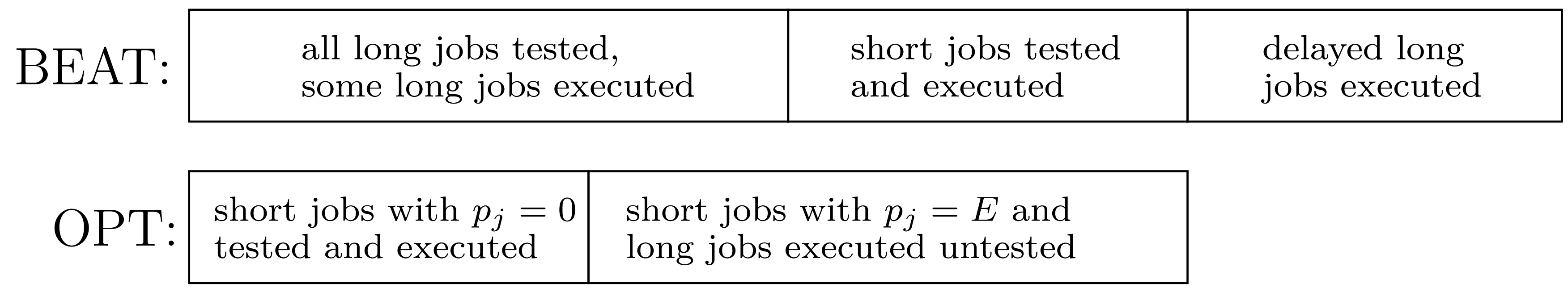
- Adversary chooses $\alpha,\beta,\gamma \geq 0$ with $\alpha+\beta+\gamma \leq 1$



- Adversary chooses (α,β,γ), s.t. $G(\alpha,\beta,\gamma,T,E)$ is a local minima
- These generate conditions on T,E $G(\alpha,\beta,\gamma, T, E) \geq 0$
- Algorithm chooses T,E satisfying all conditions and has ratio $\leq T$
- Cases: (α,β,γ) is in the polytope, one of the 3 two-dimensional facets, or one of the 6 one-dimensional facets → standard but tedious second order analysis
- Optimal T,E are roots to polynomials of degree 5

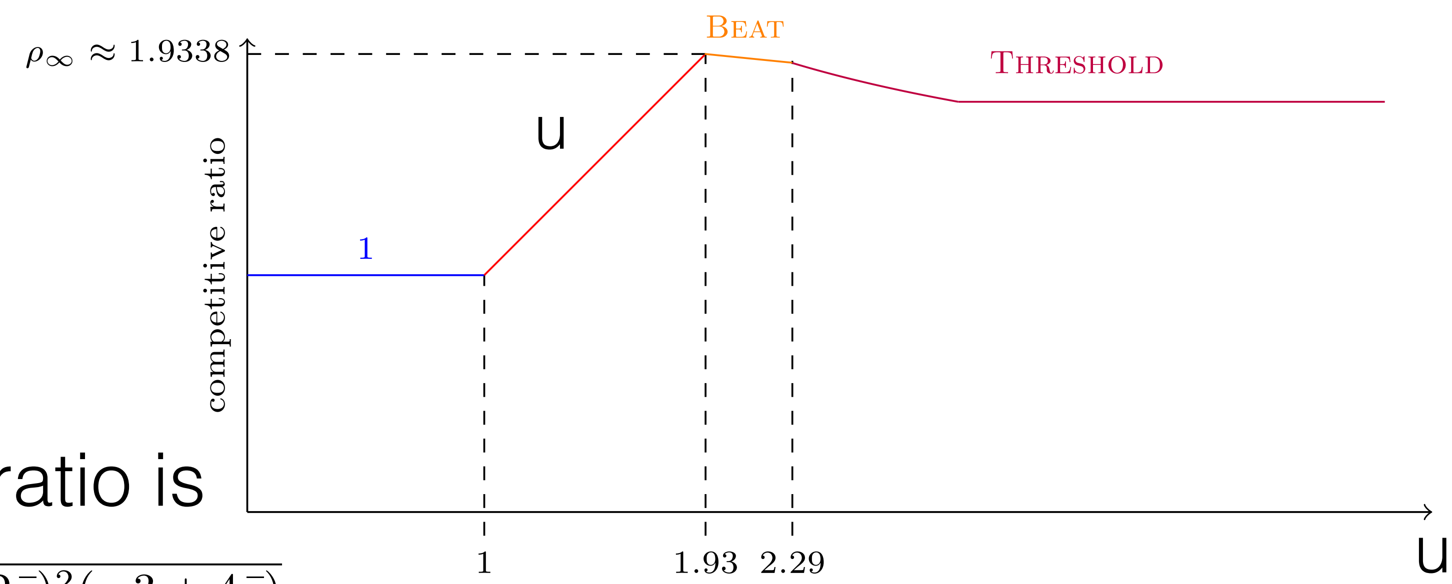


Algorithm BEAT



- n uniform jobs with upper limit u, **short** if processing time either ≤ E:=max{1,u-1}, **long** otherwise
- Algorithm: maintain TotalTest and TotalExec times.
- Test arbitrary job j and execute immediately if short or if TotalExec + p_j ≤ TotalTest
- **Worst case** instance: Essentially all jobs have processing times ∈{0,E,u}, presented in decreasing order

Algorithm BEAT



- Asymptotic competitive ratio is

$$\rho_\infty^{BEAT} = \frac{1+2(-2+\bar{p})\bar{p}+\sqrt{(1-2\bar{p})^2(-3+4\bar{p})}}{2(-1+\bar{p})\bar{p}}$$

- Algorithm: maintain TotalTest and TotalExec times.

- Test arbitrary job j and execute immediately if short or if TotalExec + $p_j \leq$ TotalTest

- **Worst case** instance: Essentially all jobs have processing times ∈{0,E,u}, presented in decreasing order

Future directions

- Is the deterministic ratio < 2 ?
- Consider test times proportional to u_j
- Study other classical combinatorial problems