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

input → compute → output
Computing paradigm

input

compute

output
Models with uncertainty

- Stochastic optimization
- Robust optimization
- Some papers with queries
  - 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
  - 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
  - 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

![Graph Diagram]

- Edges:
  - 1
  - (2,6)
  - (4,9)
  - 5
  - 3
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
  \[ \text{ALG}_I \leq c \text{ OPT}_I \]

• 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**: is 1
- **ALG**: is n-1
- **Ratio**: terrible large
## Minimum spanning tree

<table>
<thead>
<tr>
<th></th>
<th>competitive ratio</th>
<th>lower bound</th>
<th>upper bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>deterministic</td>
<td>[Erlebach et al. STACS’2008]</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>randomized</td>
<td>[Megow, Meißner, Skutella, ESA’2015]</td>
<td>1.5</td>
<td>1.707</td>
</tr>
</tbody>
</table>
### Minimum spanning tree

<table>
<thead>
<tr>
<th>Competitive ratio</th>
<th>Lower bound</th>
<th>Upper bound</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deterministic</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Erlebach et al. STACS'2008]</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td><strong>Randomized</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Megow, Meißner, Skutella, ESA'2015]</td>
<td>1.5</td>
<td>1.707</td>
</tr>
</tbody>
</table>
The witness algorithm

• For a more general setting: **Cheapest Set Problem**

• Find a *feasible* set $S \subseteq \{1, \ldots, n\}$ minimizing $\sum_{i \in S} x_i$

• $W$ 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 $\leq c$, 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 $e$ where $w_e \in (L_e, U_e)$

- **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 $T=\emptyset$
  
  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

could compress it before sending

cares about the reception time
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 \( 0 \leq p \leq u \), and schedule it:
  - cost \( 1 + p \)

\( u \) is given
\( p \) is hidden
A test reveals \( p \)
Minimize competitive ratio

• 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 \( u = \phi = \text{golden ratio} = 1.618... \) (satisfies \( \phi + 1 = \phi^2 \))

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

- **If** algorithm does test, adversary chooses \( p = \phi \) and does not test
The general problem

has to send files of various sizes

could compress files before sending

cares about $\sum C_j$

$C_j =$ reception time of file $j$
### Other motivations

<table>
<thead>
<tr>
<th><strong>Code optimizer</strong></th>
<th><strong>safe problem resolution versus heuristic</strong></th>
<th><strong>Scheduling medical appointments</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine could run a code optimizer before executing a program</td>
<td>There are two methods to solve a problem. A safe one and a heuristic that might be quicker or fail</td>
<td>Quick diagnosis can estimate processing times</td>
</tr>
</tbody>
</table>
The general problem

- **Input**: n jobs with upper limits $u_1, ..., u_n$

- **Produce** a schedule consisting of job executions or tests. Test of job j takes 1 time unit and changes its processing time to $0 \leq p_j \leq u_j$. 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

<table>
<thead>
<tr>
<th>competitive ratio</th>
<th>lower bound</th>
<th>upper bound</th>
<th>algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>deterministic ratio</strong></td>
<td>1.8546</td>
<td>2</td>
<td>THRESHOLD</td>
</tr>
<tr>
<td><strong>randomized ratio</strong></td>
<td>1.6257</td>
<td>1.7453</td>
<td>RANDOM</td>
</tr>
<tr>
<td><strong>det. ratio. on uniform instances (u_j=p)</strong></td>
<td>1.8546</td>
<td>1.9338</td>
<td>BEAT</td>
</tr>
<tr>
<td><strong>det. ratio. on extreme uniform instances (u_j=p, p_j∈{0,p})</strong></td>
<td>1.8546</td>
<td>1.8668</td>
<td>UTE</td>
</tr>
<tr>
<td><strong>det. ratio on extreme uniform instances with u=1.9896</strong></td>
<td>1.8546</td>
<td>1.8552</td>
<td>UTE</td>
</tr>
</tbody>
</table>
Our results

<table>
<thead>
<tr>
<th>competitive ratio</th>
<th>lower bound</th>
<th>upper bound</th>
<th>algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>deterministic ratio</td>
<td>1.8546</td>
<td>2</td>
<td>THRESHOLD</td>
</tr>
<tr>
<td>randomized ratio</td>
<td>1.6257</td>
<td>1.7453 (asymptotic ratio)</td>
<td>RANDOM</td>
</tr>
<tr>
<td>det. ratio. on uniform instances (uj=p)</td>
<td>1.8546</td>
<td>1.9338</td>
<td>BEAT</td>
</tr>
<tr>
<td>det. ratio. on extreme uniform instances (uj=p, pj∈{0,p})</td>
<td>1.8546</td>
<td>1.8668</td>
<td>UTE</td>
</tr>
<tr>
<td>det. ratio on extreme uniform instances with u=1.9896</td>
<td>1.8546</td>
<td>1.8552</td>
<td>UTE</td>
</tr>
</tbody>
</table>
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 $j \geq \delta n$ or job $j$ is executed untested by algo.
  $p_j = p$ otherwise
- Algorithm gets even to know $\delta$
- Any decent algorithm produces a schedule with above structure for parameters $\nu, \lambda$ with $\nu + \lambda \leq \delta$
- The competitive ratio is $\frac{\text{ALG}(\delta, \nu, \lambda, n)}{\text{OPT}(\delta, \nu, n)}$
- Algorithm (minimizer) chooses $\nu, \lambda$
- Adversary (maximizer) chooses $n, \delta$
- Analyzing local optima yields ratio 1.854628
## Our results

<table>
<thead>
<tr>
<th>competitive ratio</th>
<th>lower bound</th>
<th>upper bound</th>
<th>algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>deterministic ratio</td>
<td>1.8546</td>
<td>2</td>
<td>THRESHOLD</td>
</tr>
<tr>
<td>randomized ratio</td>
<td>1.6257</td>
<td>1.7453</td>
<td>RANDOM</td>
</tr>
<tr>
<td>det. ratio. on uniform instances ((u_j=p))</td>
<td>1.8546</td>
<td>1.9338</td>
<td>BEAT</td>
</tr>
<tr>
<td>det. ratio. on extreme uniform instances ((u_j=p, p_j\in{0,p}))</td>
<td>1.8546</td>
<td>1.8668</td>
<td>UTE</td>
</tr>
<tr>
<td>det. ratio on extreme uniform instances with (u=1.9896)</td>
<td>1.8546</td>
<td>1.8552</td>
<td>UTE</td>
</tr>
</tbody>
</table>
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:
  \[ \text{ALG}(a,b,c) \leq 2 \cdot \text{OPT}(a,b,c) \]
Our results

<table>
<thead>
<tr>
<th>competitive ratio</th>
<th>lower bound</th>
<th>upper bound</th>
<th>algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>deterministic ratio</td>
<td>1.8546</td>
<td>2</td>
<td>THRESHOLD</td>
</tr>
<tr>
<td>randomized ratio</td>
<td>1.6257</td>
<td>1.7453 (asymptotic ratio)</td>
<td>RANDOM</td>
</tr>
<tr>
<td>det. ratio. on uniform instances ($u_j=p$)</td>
<td>1.8546</td>
<td>1.9338</td>
<td>BEAT</td>
</tr>
<tr>
<td>det. ratio. on extreme uniform instances ($u_j=p$, $p_j\in{0,p}$)</td>
<td>1.8546</td>
<td>1.8668</td>
<td>UTE</td>
</tr>
<tr>
<td>det. ratio on extreme uniform instances with $u=1.9896$</td>
<td>1.8546</td>
<td>1.8552</td>
<td>UTE</td>
</tr>
</tbody>
</table>
**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 $\beta = \frac{1 - \bar p + \bar p^2 - \rho + 2\bar pp - \bar p^2 \rho}{1 - \bar p + \bar p^2 - \rho + \bar pp}$
- Execute all jobs untested if $p \leq \rho$
- Otherwise test all jobs. Execute right after their test the first $\max\{0,\beta\}$ fraction of jobs. Then only if $p_j=0$. Finally execute deferred jobs.
- **Worst case** instance defined by length $p$ fraction $\gamma$: the first $\gamma n$ tested jobs have $p_j=p$ and the remaining $p_j=0$
- Second order analysis to optimize $p, \gamma$ and $\beta$
Our results

<table>
<thead>
<tr>
<th>competitive ratio</th>
<th>lower bound</th>
<th>upper bound</th>
<th>algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>deterministic ratio</td>
<td>1.8546</td>
<td>2</td>
<td>THRESHOLD</td>
</tr>
<tr>
<td>randomized ratio</td>
<td>1.6257</td>
<td>1.7453 (asymptotic ratio)</td>
<td>RANDOM</td>
</tr>
<tr>
<td>det. ratio. on uniform instances ((u_j=p))</td>
<td>1.8546</td>
<td>1.9338</td>
<td>BEAT</td>
</tr>
<tr>
<td>det. ratio. on extreme uniform instances ((u_j=p, p_j \in {0,p}))</td>
<td>1.8546</td>
<td>1.8668</td>
<td>UTE</td>
</tr>
<tr>
<td>det. ratio on extreme uniform instances with (u=1.9896)</td>
<td>1.8546</td>
<td>1.8552</td>
<td>UTE</td>
</tr>
</tbody>
</table>
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:**
- $(1-\alpha-\beta-\gamma)$ fraction of jobs: $u_j=T$, $p_j=0$ (not mentioned in the original content)
- $\alpha n$ jobs have $u_j=T$, $p_j=T$
- $\beta n$ jobs have $u_j=E$, $p_j=E$
- $\gamma n$ jobs have $u_j=E+\varepsilon$, $p_j=E+\varepsilon$

**Ratio $\leq T$ iff**
- $G := \text{OPT} \cdot T - \text{ALG} \geq 0$

**Algorithm chooses $T$, $E$ to maximize $G$**

Adversary chooses $\alpha, \beta, \gamma$ to minimize $G$.
Our results

<table>
<thead>
<tr>
<th>competitive ratio</th>
<th>lower bound</th>
<th>upper bound</th>
<th>algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>deterministic ratio</td>
<td>1.8546</td>
<td>2</td>
<td>THRESHOLD</td>
</tr>
<tr>
<td>randomized ratio</td>
<td>1.6257</td>
<td>1.7453 (asymptotic ratio)</td>
<td>RANDOM</td>
</tr>
<tr>
<td>det. ratio. on uniform instances (u_j=p)</td>
<td>1.8546</td>
<td>1.9338</td>
<td>BEAT</td>
</tr>
<tr>
<td>det. ratio. on extreme uniform instances (u_j=p, p_j∈{0,p})</td>
<td>1.8546</td>
<td>1.8668</td>
<td>UTE</td>
</tr>
<tr>
<td>det. ratio on extreme uniform instances with u=1.9896</td>
<td>1.8546</td>
<td>1.8552</td>
<td>UTE</td>
</tr>
</tbody>
</table>
Algorithm RANDOM

- Adversary chooses $\alpha, \beta, \gamma \geq 0$ with $\alpha + \beta + \gamma \leq 1$
- Adversary chooses $(\alpha, \beta, \gamma)$, s.t. $G(\alpha, \beta, \gamma, T, E)$ is a local minima
- These generate conditions on $T, E$
- Algorithm chooses $T, E$ satisfying all conditions and has ratio $\leq T$
- Cases: $(\alpha, \beta, \gamma)$ 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

fractions: $1-\alpha-\beta-\gamma$ $\alpha$ $\beta$ $\gamma$ $\gamma$

ALG: $\begin{bmatrix} 1 & 1 & T & 1 & E & 1 & E+\varepsilon & E+\varepsilon & E+\varepsilon \end{bmatrix}$

tests in random order
deferred jobs

OPT: $\begin{bmatrix} 1 & 1 & 1 & 1 & T & T & T & E & E & E & E+\varepsilon & E+\varepsilon & E+\varepsilon \end{bmatrix}$
Algorithm BEAT

<table>
<thead>
<tr>
<th>BEAT:</th>
<th>all long jobs tested, some long jobs executed</th>
<th>short jobs tested and executed</th>
<th>delayed long jobs executed</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPT:</td>
<td>short jobs with ( p_j = 0 ) tested and executed</td>
<td>short jobs with ( p_j = E ) and long jobs executed untested</td>
<td></td>
</tr>
</tbody>
</table>

- \( n \) uniform jobs with upper limit \( u \), **short** if processing time either \( \leq E := \max\{1, u-1\} \), **long** otherwise

- Algorithm: maintain TotalTest and TotalExec times.

- Test arbitrary job \( j \) and execute immediately if short or if \( \text{TotalExec} + p_j \leq \text{TotalTest} \)

- **Worst case** instance: Essentially all jobs have processing times \( \in \{0, E, u\} \), presented in decreasing order
Algorithm BEAT

- Asymptotic competitive ratio is
  \[ \rho^\text{BEAT}_\infty = \frac{1 + 2(-2 + \bar{p})\bar{p} + \sqrt{(1 - 2\bar{p})^2(-3 + 4\bar{p})}}{2(-1 + \bar{p})\bar{p}} \]

- 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 \( \in \{0, E, u\} \), presented in decreasing order
Future directions

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