A Survey on Advice and Randomization for the Knapsack Problem

Hans-Joachim Böckenhauer, Dennis Komm, Richard Královič, and Peter Rossmanith

July 7, 2014 TOLA 2014 Copenhagen

Online Problems and Advice Complexity
Introduction

Online Problem

- Sequence of requests
- Answer each request before the next one arrives
- Minimize cost / maximize gain
- Examples: Ski Rental, k-Server, Paging, Job Shop Scheduling

Online Problem

- Sequence of requests
- Answer each request before the next one arrives
- Minimize cost / maximize gain
- Examples: Ski Rental, k-Server, Paging, Job Shop Scheduling

Competitive Ratio (Maximization)

An online algorithm ALG is c-competitive if there is a constant α such that, for every instance I, we have

$$c \cdot (\text{Gain of ALG's solution on } I) + \alpha \geq \text{Optimal gain for } I$$

Online Problem

- Sequence of requests
- Answer each request before the next one arrives
- Minimize cost / maximize gain
- Examples: Ski Rental, k-Server, Paging, Job Shop Scheduling

Competitive Ratio (Maximization)

An online algorithm ALG is c-competitive if there is a constant α such that, for every instance I, we have

$$c \cdot (\text{Gain of ALG's solution on } I) + \alpha \geq \text{Optimal gain for } I$$

If optimum is constant, $\alpha = 0$.

How much information are we missing ...

- to be optimal?
- to achieve some competitive ratio?

How much information are we missing . . .

- to be optimal?
- to achieve some competitive ratio?

Example: Ski Rental

How much information are we missing ...

- to be optimal?
- to achieve some competitive ratio?

Example: Ski Rental

- No information about future ⇨ 2-competitive

How much information are we missing . . .

- to be optimal?
- to achieve some competitive ratio?

Example: Ski Rental

- No information about future ⇨ 2-competitive
- One bit of advice ⇨ optimal

How much information are we missing ...

- to be optimal?
- to achieve some competitive ratio?

Example: Ski Rental

- No information about future ⇨ 2-competitive
- One bit of advice ⇨ optimal

Motivation

- Theoretical interest: Deeper understanding of the problems
- "Essence" of the problem
- Bounds for randomization

The Model

Computation with Advice

Oracle with unlimited power:

1. Sees all requests
2. Prepares infinite tape

The Model

Computation with Advice

Oracle with unlimited power:

1. Sees all requests
2. Prepares infinite tape

Algorithm starts:

3. Processes n requests one by one, can use advice tape
4. Advice: Total number of advice bits accessed

The Model

Computation with Advice

Oracle with unlimited power:

1. Sees all requests
2. Prepares infinite tape

Algorithm starts:

3. Processes n requests one by one, can use advice tape
4. Advice: Total number of advice bits accessed

Analysis

- Solution: (oracle, algorithm)
- Correctness: the pair works correctly on all inputs
- Advice complexity: Maximal advice over all inputs of length $\leq n$

The Knapsack Problem

Introduction

Definition (KP)

Given a knapsack of weight capacity 1, n objects arrive in successive time steps where each object has

- a weight $w_i \leq 1$
- a value v_i

Maximize value of packed objects without exceeding capacity

After an object is offered, it must be specified whether it is part of the solution

Definition (KP)

Given a knapsack of weight capacity 1, n objects arrive in successive time steps where each object has

- a weight $w_i \leq 1$
- a value v_i

Maximize value of packed objects without exceeding capacity

After an object is offered, it must be specified whether it is part of the solution

In its **simple** version (SKP), $w_i = v_i$, for every object i

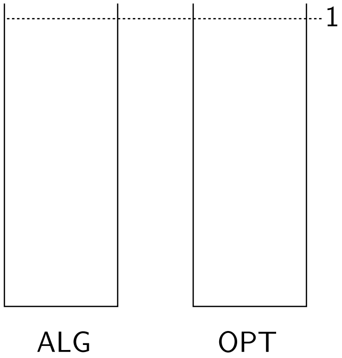
Theorem (Marchetti-Spaccamela and Vercellis; 1995)

Any deterministic online algorithm has an arbitrarily large competitive ratio for the SKP (and thus KP).

Theorem (Marchetti-Spaccamela and Vercellis; 1995)

Any deterministic online algorithm has an arbitrarily large competitive ratio for the SKP (and thus KP).

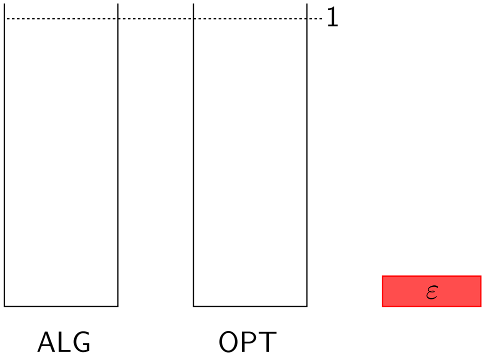
Let $\varepsilon > 0$



Theorem (Marchetti-Spaccamela and Vercellis; 1995)

Any deterministic online algorithm has an arbitrarily large competitive ratio for the SKP (and thus KP).

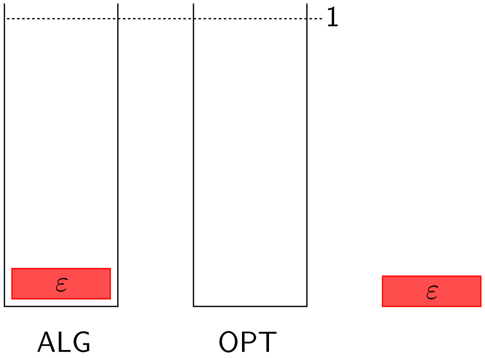
Let $\varepsilon > 0$



Theorem (Marchetti-Spaccamela and Vercellis; 1995)

Any deterministic online algorithm has an arbitrarily large competitive ratio for the SKP (and thus KP).

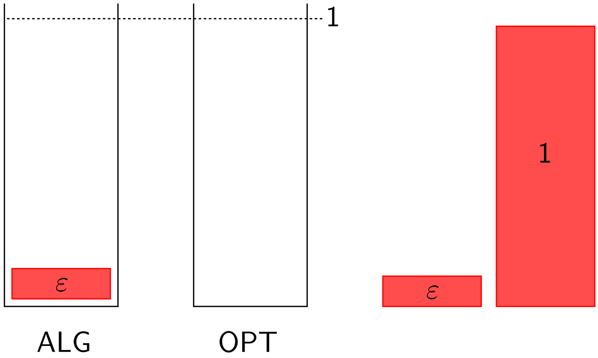
Let $\varepsilon > 0$



Theorem (Marchetti-Spaccamela and Vercellis; 1995)

Any deterministic online algorithm has an arbitrarily large competitive ratio for the SKP (and thus KP).

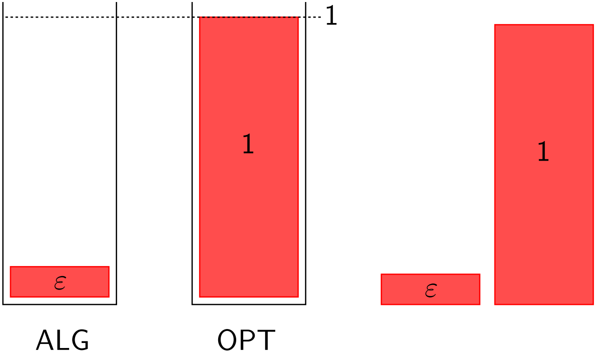
Let $\varepsilon > 0$



Theorem (Marchetti-Spaccamela and Vercellis; 1995)

Any deterministic online algorithm has an arbitrarily large competitive ratio for the SKP (and thus KP).

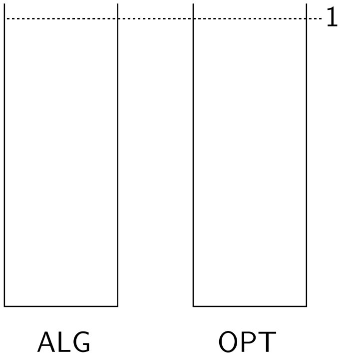
Let $\varepsilon > 0$, $\text{comp}(\text{ALG}(I)) = 1/\varepsilon$



Theorem (Marchetti-Spaccamela and Vercellis; 1995)

Any deterministic online algorithm has an arbitrarily large competitive ratio for the SKP (and thus KP).

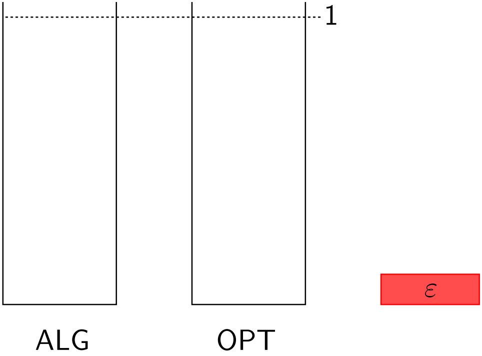
Let $\varepsilon > 0$



Theorem (Marchetti-Spaccamela and Vercellis; 1995)

Any deterministic online algorithm has an arbitrarily large competitive ratio for the SKP (and thus KP).

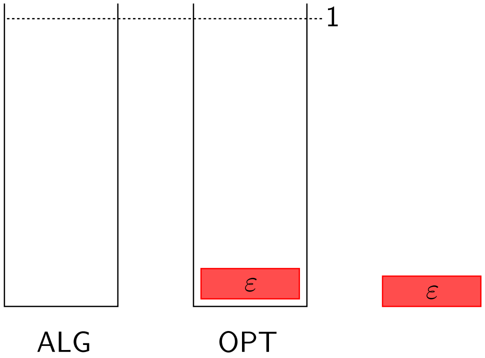
Let $\varepsilon > 0$



Theorem (Marchetti-Spaccamela and Vercellis; 1995)

Any deterministic online algorithm has an arbitrarily large competitive ratio for the SKP (and thus KP).

Let $\varepsilon > 0$, $\text{comp}(\text{ALG}(I)) = \varepsilon/0$



The Simple Knapsack Problem

Advice for Optimality

Theorem

There is an optimal online algorithm for the SKP that reads n advice bits.

Theorem

There is an optimal online algorithm for the SKP that reads n advice bits.

- With every offered object i, read one bit b_i

If $\mathbf{b_i = 0}$: `take object`

If $\mathbf{b_i = 1}$: `discard object`

Theorem

There is an optimal online algorithm for the SKP that reads n advice bits.

- With every offered object i, read one bit b_i

If $\mathbf{b_i = 0}$: `take object`
If $\mathbf{b_i = 1}$: `discard object`

- This bound is tight. In other words, ...

Theorem

Any online algorithm with advice for the SKP needs to use at least $n-1$ advice bits to be optimal.

The Simple Knapsack Problem

Small Advice

Observation

- If sum of weights ≤ 1, greedy is optimal
- Else if every object has weight $< \delta$, gain of greedy is $\geq 1 - \delta$

Observation

- If sum of weights ≤ 1, greedy is optimal
- Else if every object has weight $< \delta$, gain of greedy is $\geq 1 - \delta$

Theorem

There is an online algorithm ALG *for the SKP that uses* 1 *advice bit and that is* 2*-competitive.*

Observation

- If sum of weights ≤ 1, greedy is optimal
- Else if every object has weight $< \delta$, gain of greedy is $\geq 1 - \delta$

Theorem

There is an online algorithm ALG *for the SKP that uses* 1 *advice bit and that is* 2*-competitive.*

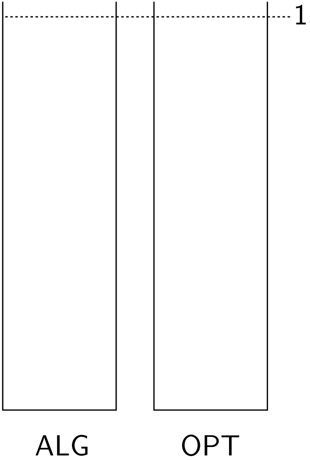
- Oracle writes 1 on tape iff one object has size $> 1/2$
- Consider ALG that reads one bit *b* of advice

If b = 0: `satisfy greedily`

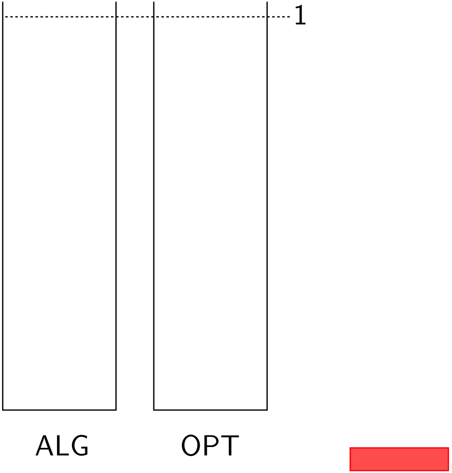
If b = 1: `wait for object of size` $> 1/2$

Case 1: $b = 0$

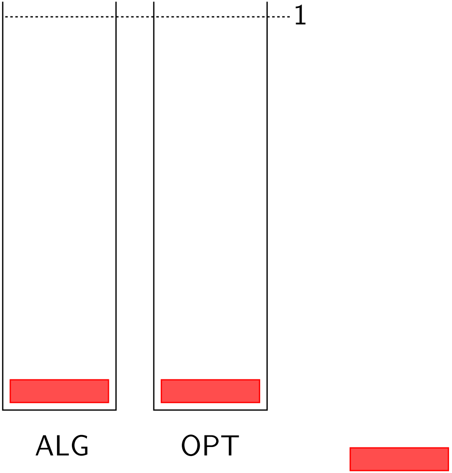
Case 1: $b = 0$



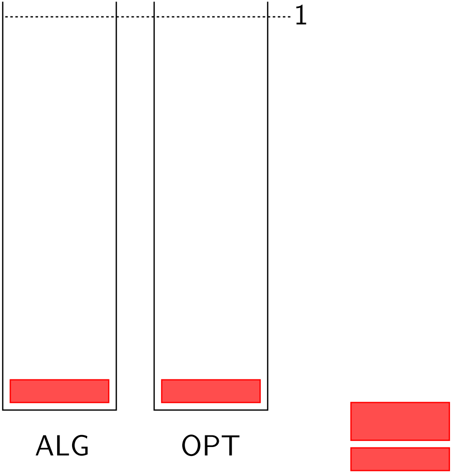
Case 1: $b = 0$



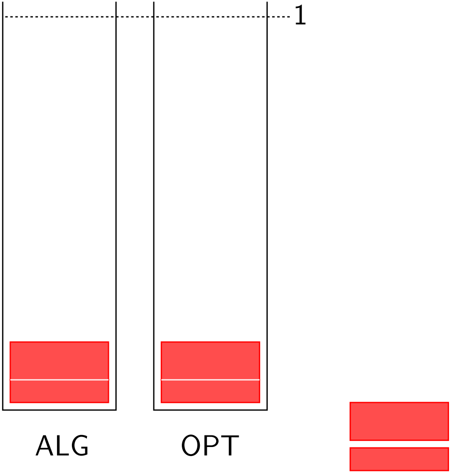
Case 1: $b = 0$



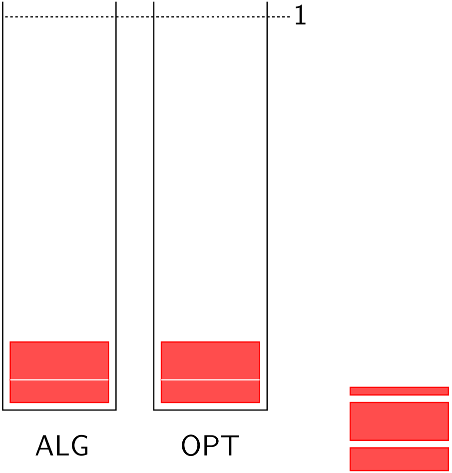
Case 1: $b = 0$



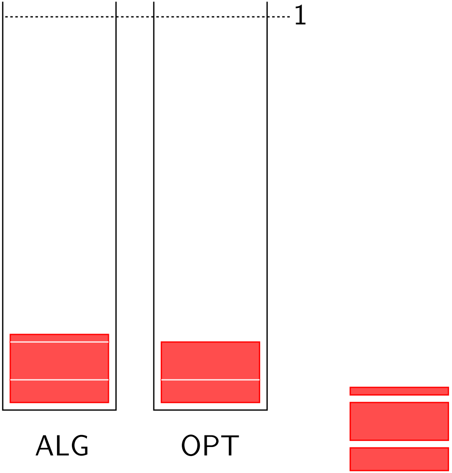
Case 1: $b = 0$



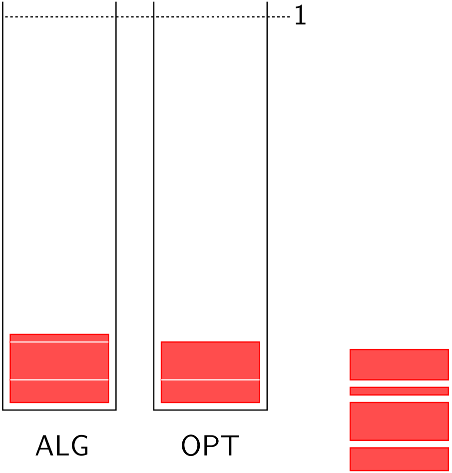
Case 1: $b = 0$



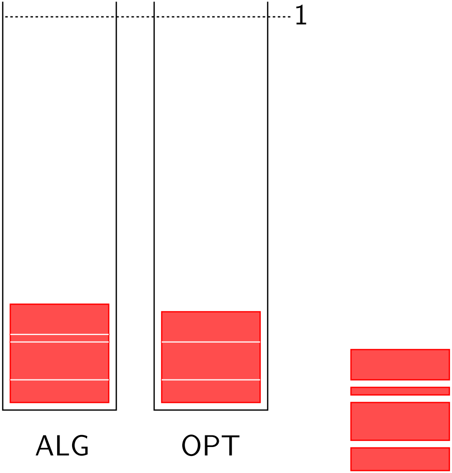
Case 1: $b = 0$



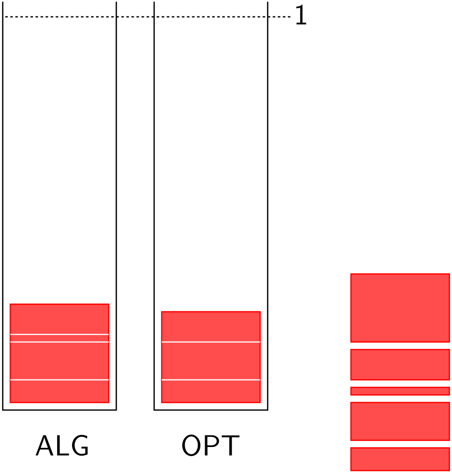
Case 1: $b = 0$



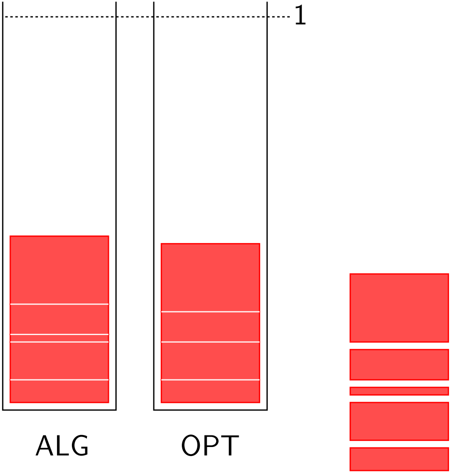
Case 1: $b = 0$



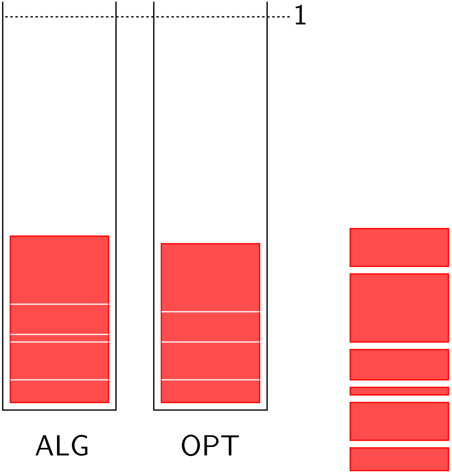
Case 1: $b = 0$



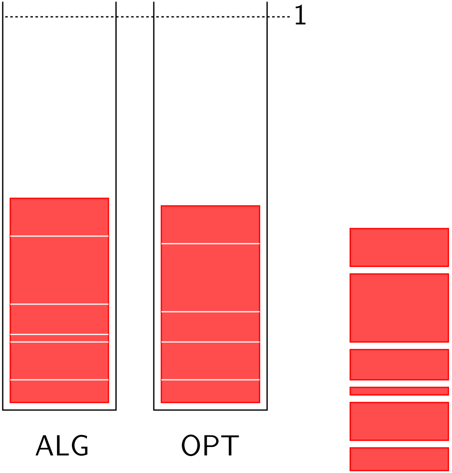
Case 1: $b = 0$



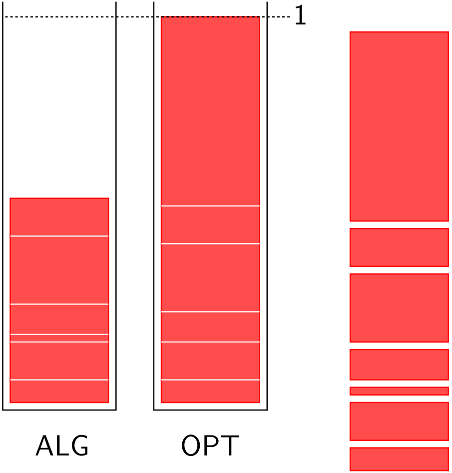
Case 1: $b = 0$



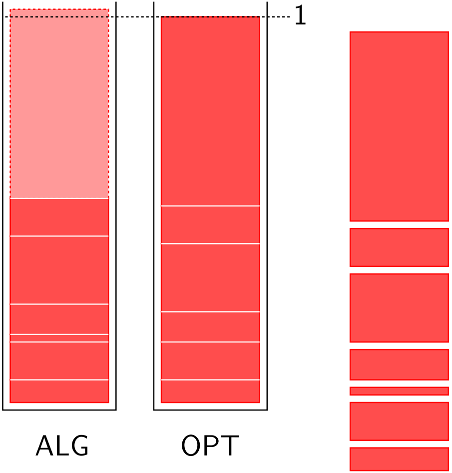
Case 1: $b = 0$



Case 1: $b = 0$

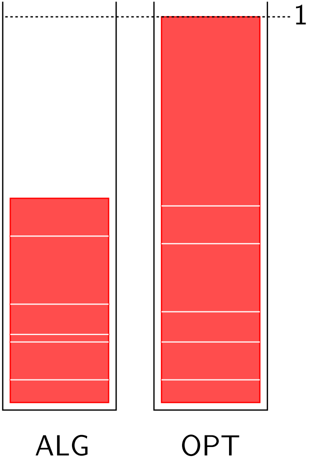


Case 1: $b = 0$

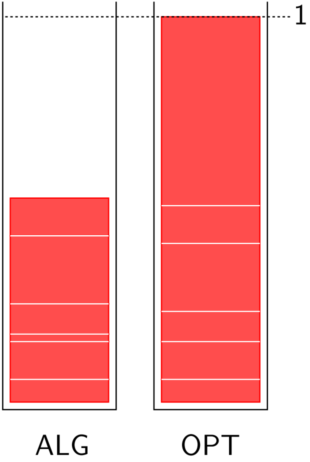


Case 1: $b = 0$

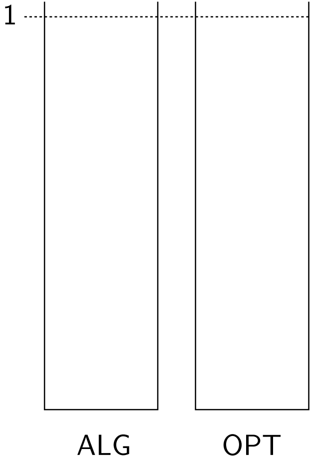
Case 2: $b = 1$



Case 1: $b = 0$

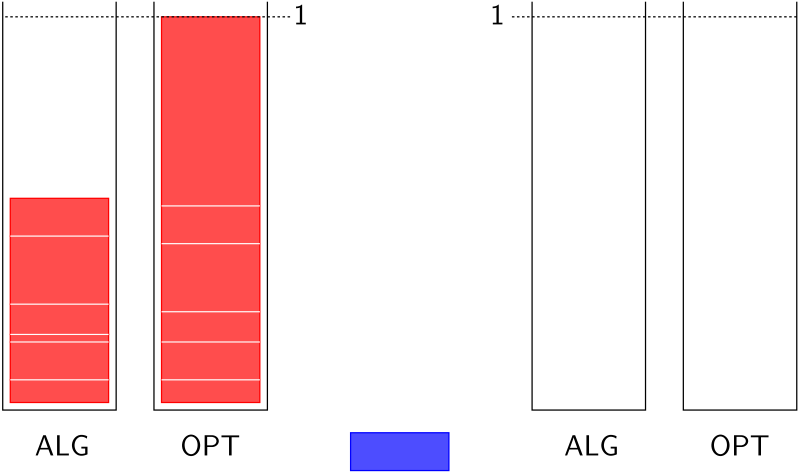


Case 2: $b = 1$



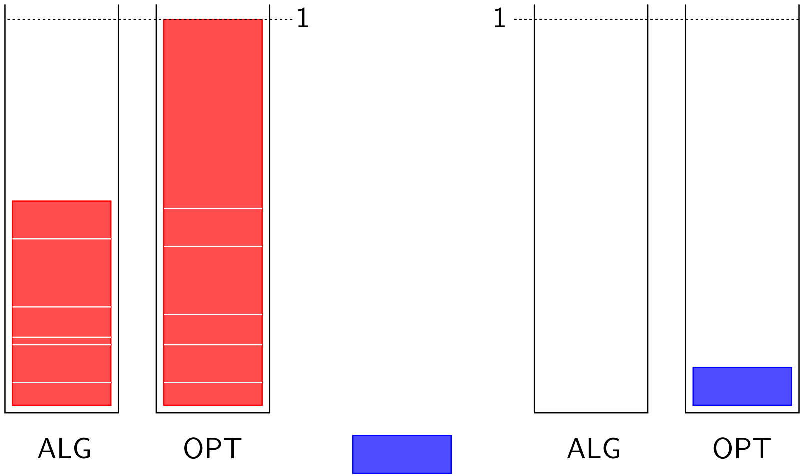
The Simple Knapsack Problem – Small Advice

Case 1: $b = 0$ Case 2: $b = 1$

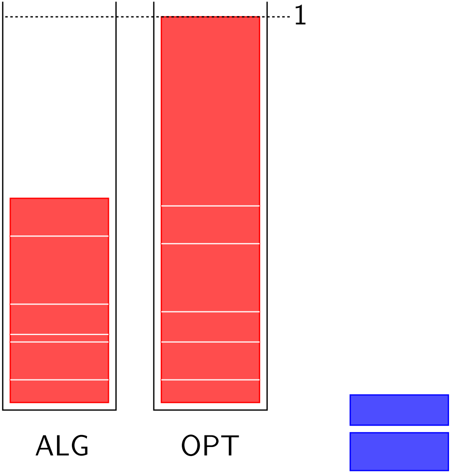


Case 1: $b = 0$

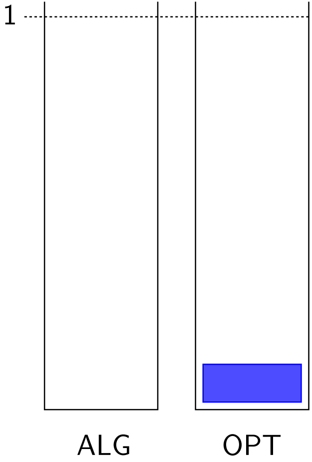
Case 2: $b = 1$



Case 1: $b = 0$



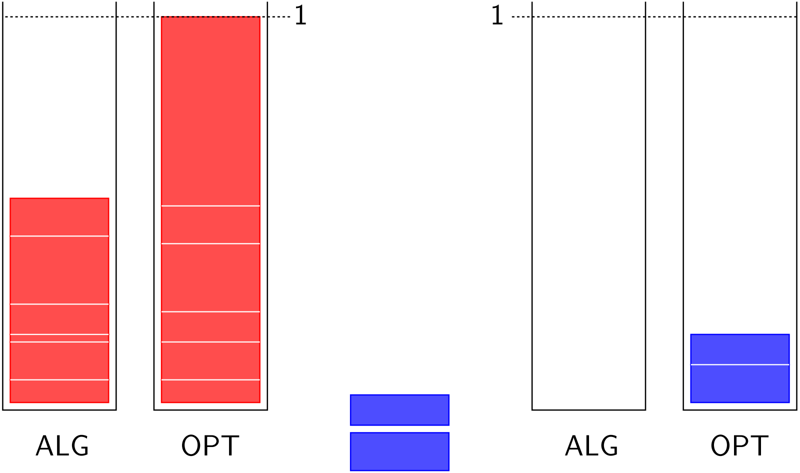
Case 2: $b = 1$



The Simple Knapsack Problem – Small Advice

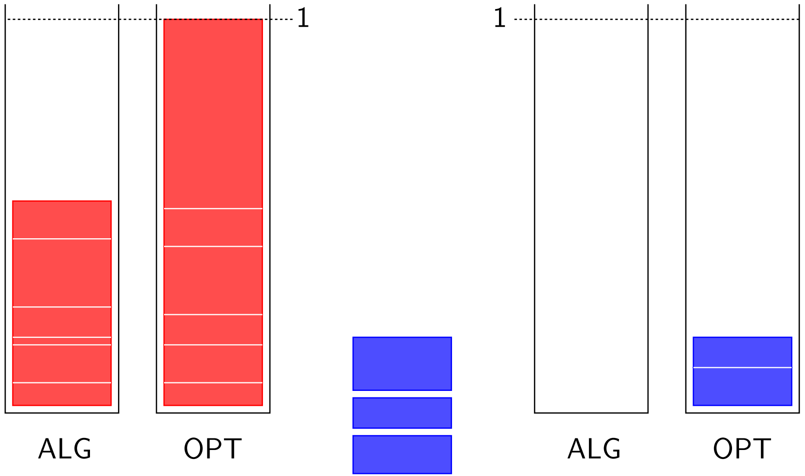
Case 1: $b = 0$

Case 2: $b = 1$



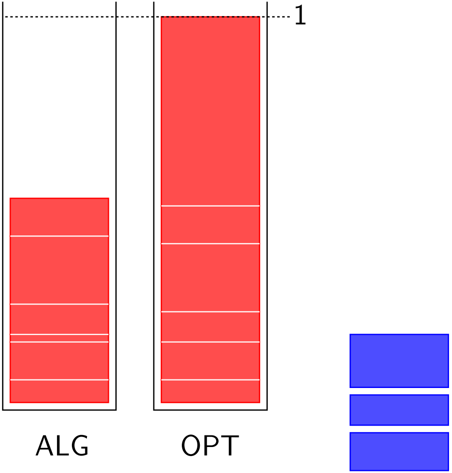
The Simple Knapsack Problem – Small Advice

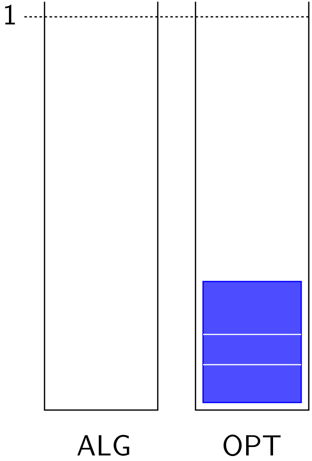
Case 1: $b = 0$ Case 2: $b = 1$



Case 1: $b = 0$

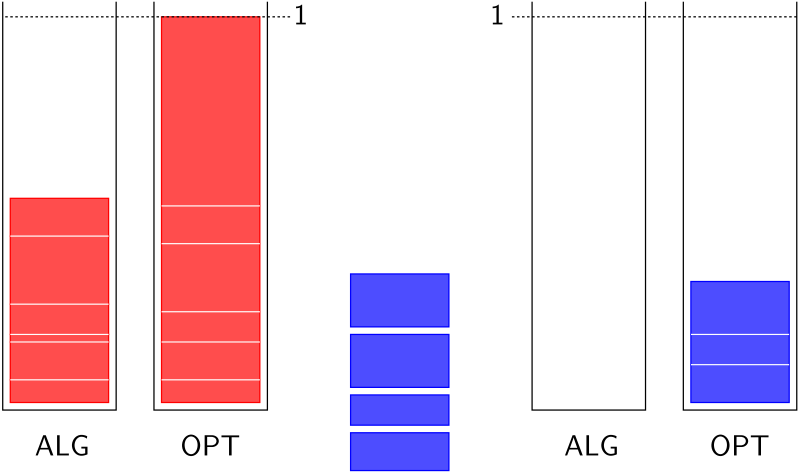
Case 2: $b = 1$





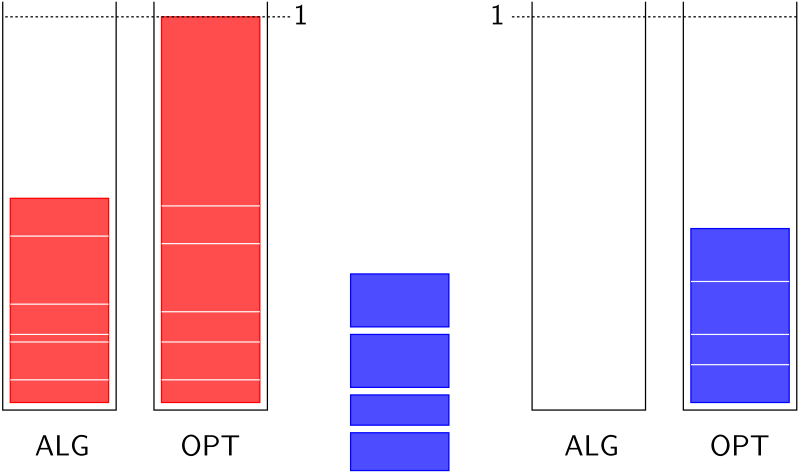
Case 1: $b = 0$

Case 2: $b = 1$



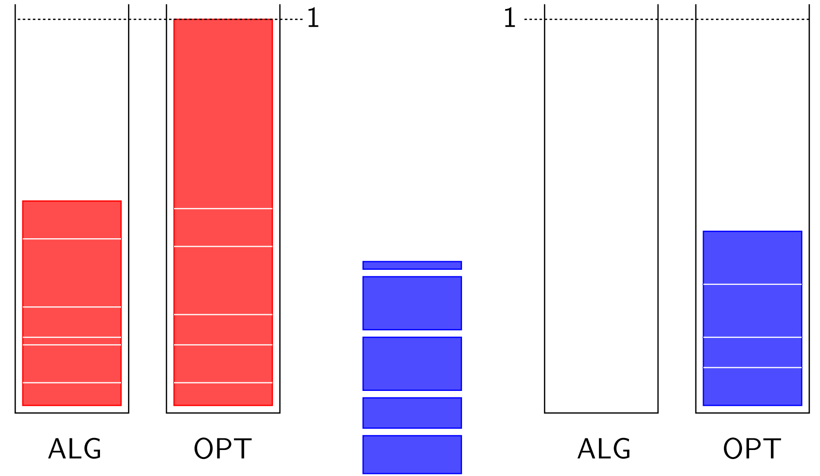
Case 1: $b = 0$

Case 2: $b = 1$

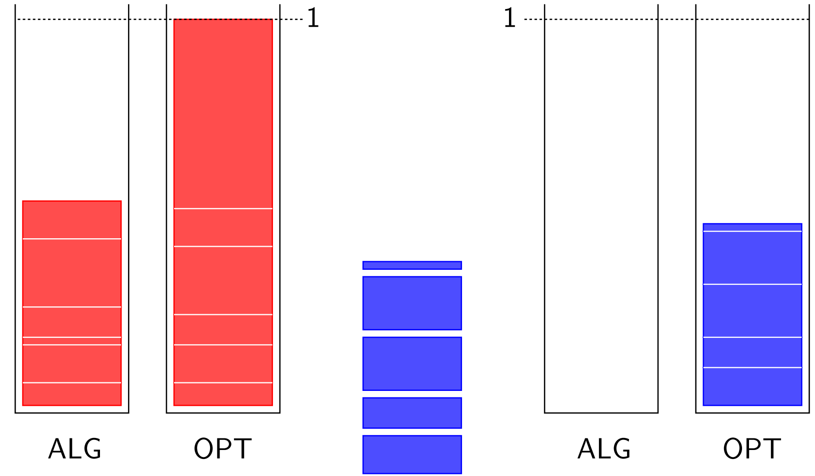


Case 1: $b = 0$

Case 2: $b = 1$

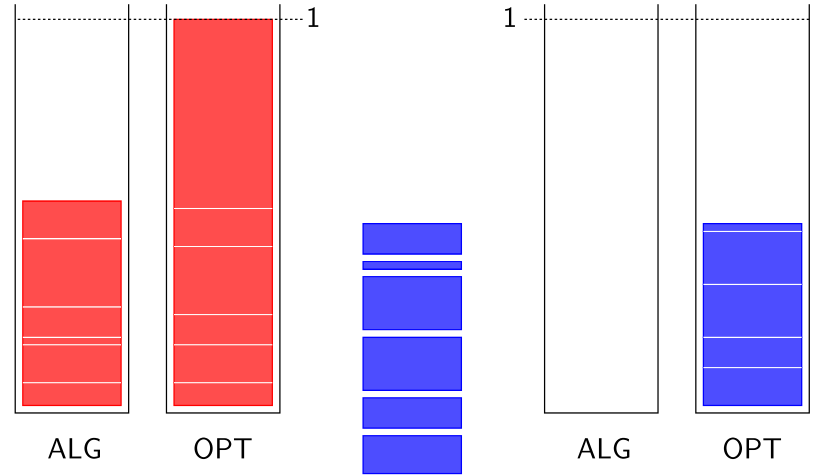


Case 1: $b = 0$ Case 2: $b = 1$



Case 1: $b = 0$

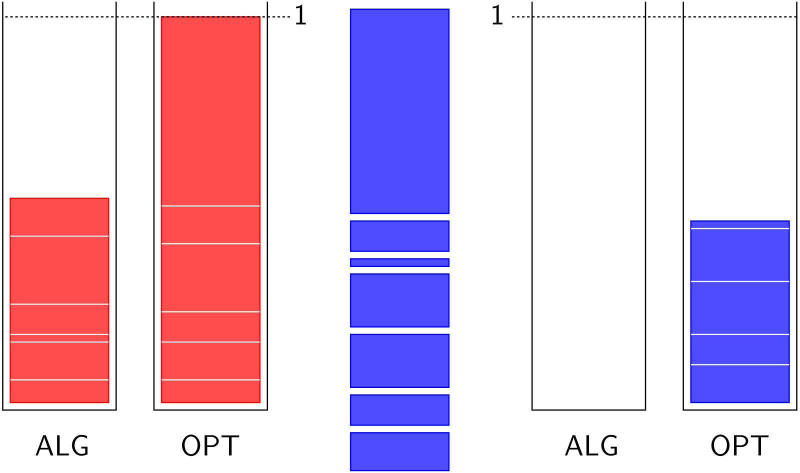
Case 2: $b = 1$



The Simple Knapsack Problem – Small Advice

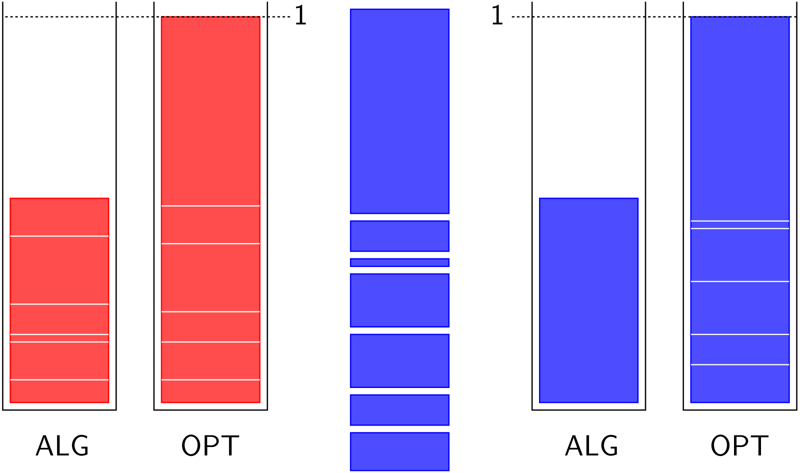
Case 1: $b = 0$

Case 2: $b = 1$



The Simple Knapsack Problem – Small Advice

Case 1: $b = 0$ Case 2: $b = 1$



Surprisingly, any additional bit **does not help** until logarithmic threshold

Theorem

No online algorithm for the SKP that uses less than $\log_2 n - 1$ advice bits is better than 2-competitive.

Surprisingly, any additional bit **does not help** until logarithmic threshold

Theorem

No online algorithm for the SKP that uses less than $\log_2 n - 1$ advice bits is better than 2-competitive.

Theorem

There is an online algorithm for the SKP that is $(1+\varepsilon)$-competitive and that uses $\mathcal{O}(\log_2 n)$ advice bits, $\varepsilon > 0$.

Surprisingly, any additional bit **does not help** until logarithmic threshold

Theorem

No online algorithm for the SKP that uses less than $\log_2 n - 1$ advice bits is better than 2-competitive.

Theorem

There is an online algorithm for the SKP that is $(1 + \varepsilon)$-competitive and that uses $\mathcal{O}(\log_2 n)$ advice bits, $\varepsilon > 0$.

- Inspect optimal solution
- ⇨ Group objects into k **heavy** (depending on ε) and **light** ones
- Compute bound t for space filled by light objects
- Number of heavy objects and t only depend on ε

Surprisingly, any additional bit **does not help** until logarithmic threshold

Theorem

No online algorithm for the SKP that uses less than $\log_2 n - 1$ *advice bits is better than 2-competitive.*

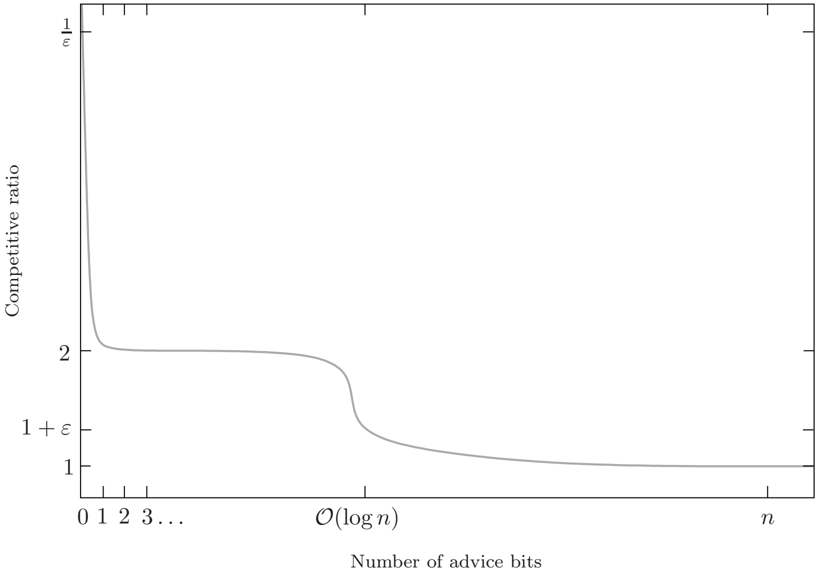
Theorem

There is an online algorithm for the SKP that is $(1+\varepsilon)$*-competitive and that uses* $\mathcal{O}(\log_2 n)$ *advice bits,* $\varepsilon > 0$*.*

- Inspect optimal solution
- ⇨ Group objects into k **heavy** (depending on ε) and **light** ones
- Compute bound t for space filled by light objects
- Number of heavy objects and t only depend on ε

However, as we have seen before, an exponential jump has to be done to be **optimal** instead of only "**very well**"

The Simple Knapsack Problem – Small Advice



The General Knapsack Problem

Small Competitive Ratio

- No longer assume that weights and values are equal and ≤ 1

⇨ Weights are ≤ 1, values are possibly larger

- No longer assume that weights and values are equal and ≤ 1
- ⇨ Weights are ≤ 1, values are possibly larger

Theorem

No online algorithm for the KP that uses less than $\log_2 n$ advice bits can obtain a competitive ratio better than 2^n.

- No longer assume that weights and values are equal and ≤ 1

⇨ Weights are ≤ 1, values are possibly larger

Theorem

No online algorithm for the KP that uses less than $\log_2 n$ advice bits can obtain a competitive ratio better than 2^n.

Theorem

There is a $(1+\varepsilon)$-competitive online algorithm for the KP that uses $\mathcal{O}(\log_2 n)$ advice bits, $\varepsilon > 0$.

- No longer assume that weights and values are equal and ≤ 1
- ⇨ Weights are ≤ 1, values are possibly larger

Theorem

No online algorithm for the KP that uses less than $\log_2 n$ advice bits can obtain a competitive ratio better than 2^n.

Theorem

There is a $(1+\varepsilon)$-competitive online algorithm for the KP that uses $\mathcal{O}(\log_2 n)$ advice bits, $\varepsilon > 0$.

- Asymptotically equivalent to simple version
- ⇨ Constant of $\mathcal{O}$ notation is much worse

Advice and Randomization

Computation with Advice

- **Oracle** ⇨ Infinite **advice** tape ⇨ Algorithm
- Oracle: Knows whole input, unlimited computational power
- Advice tape prepared before the algorithm starts
- Advice complexity $b(n)$:
  Maximal number of bits read for inputs of length n

Computation with Advice

- **Oracle** ⇨ Infinite **advice** tape ⇨ Algorithm
- Oracle: Knows whole input, unlimited computational power
- Advice tape prepared before the algorithm starts
- Advice complexity $b(n)$:
  Maximal number of bits read for inputs of length n

Randomization

- **Random source** ⇨ Infinite **random** tape ⇨ Algorithm
- Random bit complexity $r(n)$:
  Maximal number of bits read for inputs of length n

Randomization and Advice

- $2^{b(n)}$ algorithms or $2^{r(n)}$ algorithms
- Advice may be seen as best random string for every instance

⇨ Lower bounds for advice carry over to randomization

⇨ Upper bounds for randomization carry over to advice

- Small advice may lead to **barely random algorithms**, e. g.,
  - Paging
  - Job Shop Scheduling
  - **Knapsack**

Advice and Randomization

Barely Random Algorithm for the Simple Knapsack Problem

Guess one bit and act as with advice

⇨ 4-competitive in expt. and this is tight

Guess one bit and act as with advice

- ⇨ 4-competitive in expt. and this is tight, but. . .

Theorem

There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

Guess one bit and act as with advice

⇨ 4-competitive in expt. and this is tight, but...

Theorem

There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If $b = 0$: `act greedily`

If $b = 1$: `simulate greedy, start when greedy is done`

Guess one bit and act as with advice

⇨ 4-competitive in expt. and this is tight, but. . .

Theorem

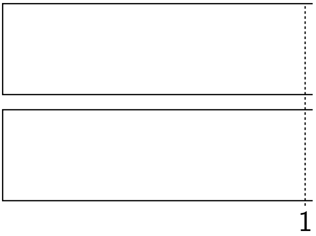
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

⇨ 4-competitive in expt. and this is tight, but...

Theorem

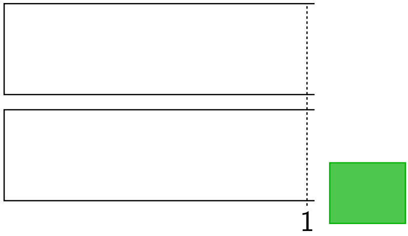
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is 2-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

⇨ 4-competitive in expt. and this is tight, but...

Theorem

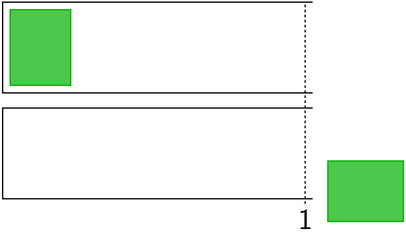
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

- 4-competitive in expt. and this is tight, but...

Theorem

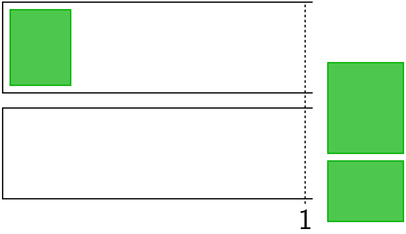
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

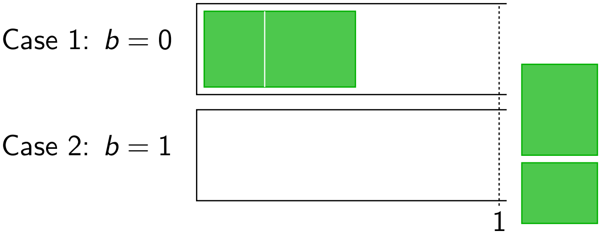
⇨ 4-competitive in expt. and this is tight, but...

Theorem

There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`



Guess one bit and act as with advice

⇨ 4-competitive in expt. and this is tight, but…

Theorem

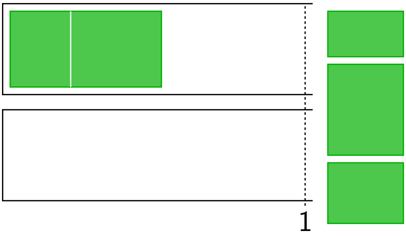
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

⇨ 4-competitive in expt. and this is tight, but...

Theorem

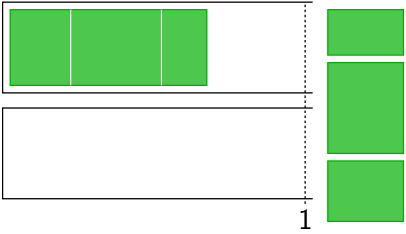
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

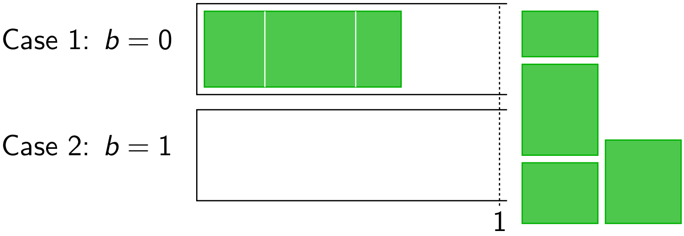
⇨ 4-competitive in expt. and this is tight, but...

Theorem

There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`



Guess one bit and act as with advice

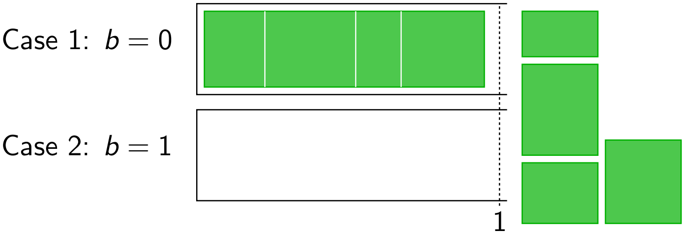
- ⇨ 4-competitive in expt. and this is tight, but. . .

Theorem

There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`



Guess one bit and act as with advice

⇨ 4-competitive in expt. and this is tight, but...

Theorem

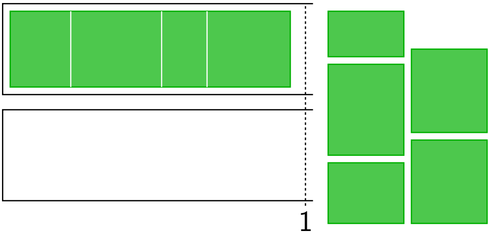
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

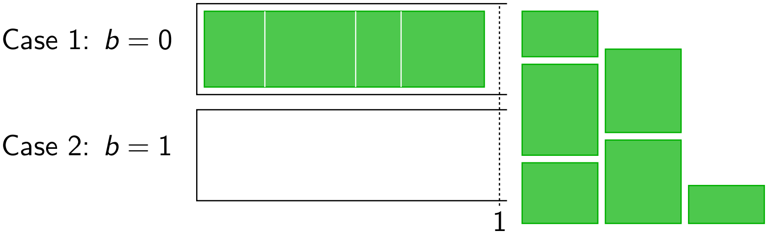
⇨ 4-competitive in expt. and this is tight, but...

Theorem

There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`



Guess one bit and act as with advice

- ⇨ 4-competitive in expt. and this is tight, but...

Theorem

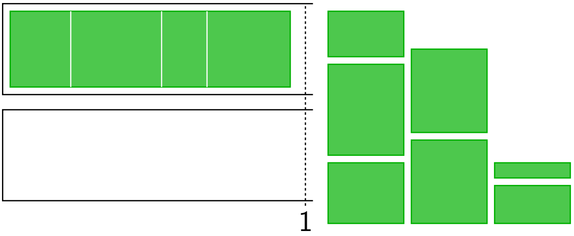
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

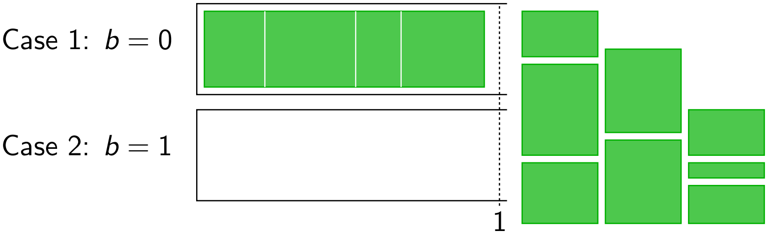
- ⇨ 4-competitive in expt. and this is tight, but...

Theorem

There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`



Guess one bit and act as with advice

⇨ 4-competitive in expt. and this is tight, but...

Theorem

There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$

Guess one bit and act as with advice

- ⇨ 4-competitive in expt. and this is tight, but. . .

Theorem

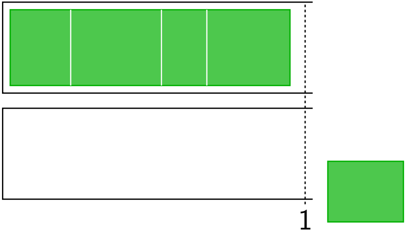
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

⇨ 4-competitive in expt. and this is tight, but. . .

Theorem

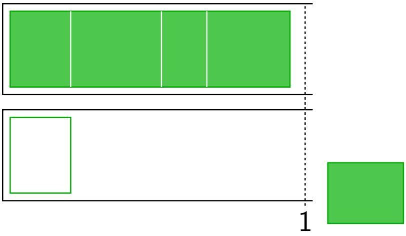
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

⇨ 4-competitive in expt. and this is tight, but...

Theorem

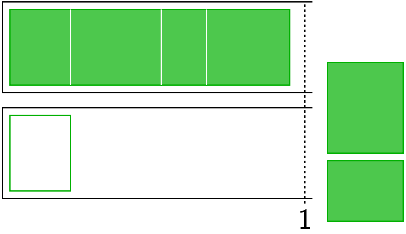
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

⇨ 4-competitive in expt. and this is tight, but…

Theorem

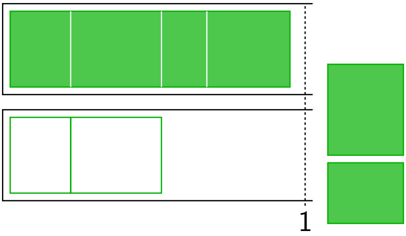
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

⇨ 4-competitive in expt. and this is tight, but. . .

Theorem

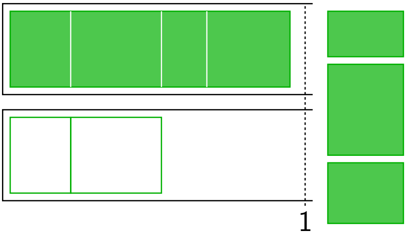
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

⇨ 4-competitive in expt. and this is tight, but...

Theorem

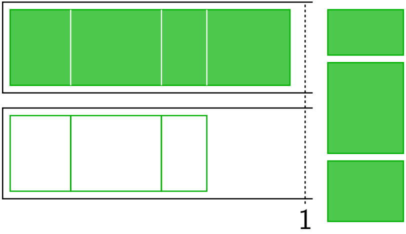
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

⇨ 4-competitive in expt. and this is tight, but…

Theorem

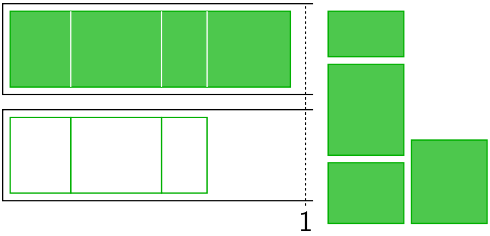
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

- ⇨ 4-competitive in expt. and this is tight, but...

Theorem

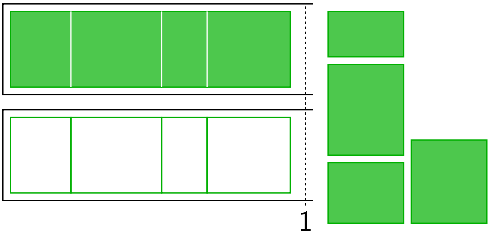
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

⇨ 4-competitive in expt. and this is tight, but. . .

Theorem

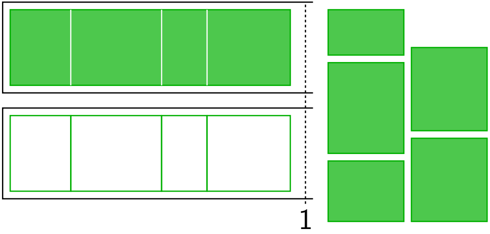
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

⇨ 4-competitive in expt. and this is tight, but...

Theorem

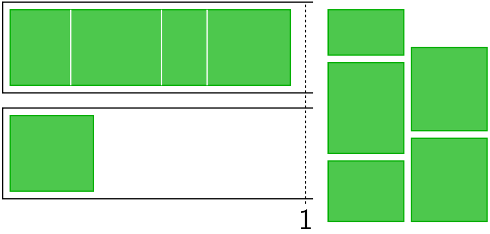
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

⇨ 4-competitive in expt. and this is tight, but...

Theorem

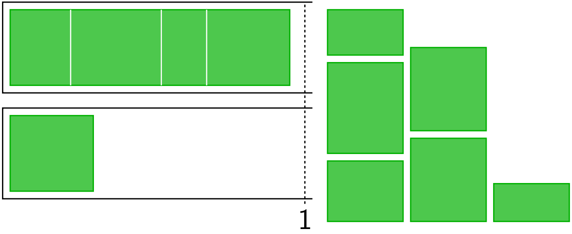
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If $b = 0$: `act greedily`

If $b = 1$: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

- ⇨ 4-competitive in expt. and this is tight, but...

Theorem

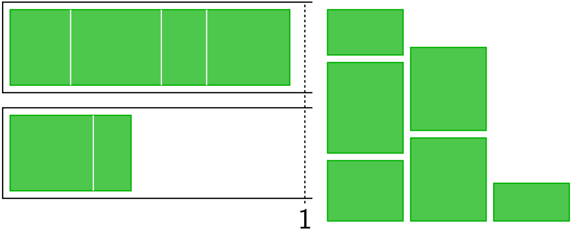
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

⇨ 4-competitive in expt. and this is tight, but…

Theorem

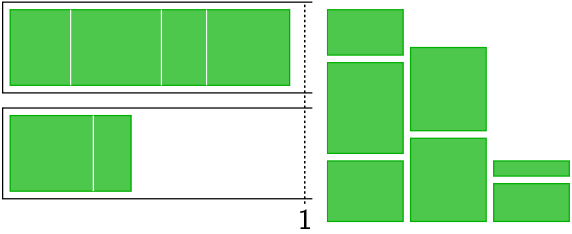
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If $b = 0$: `act greedily`

If $b = 1$: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

⇨ 4-competitive in expt. and this is tight, but...

Theorem

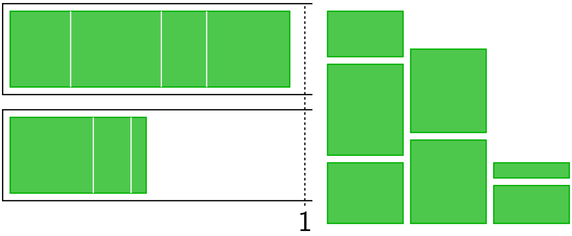
There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`

Case 1: $b = 0$

Case 2: $b = 1$



Guess one bit and act as with advice

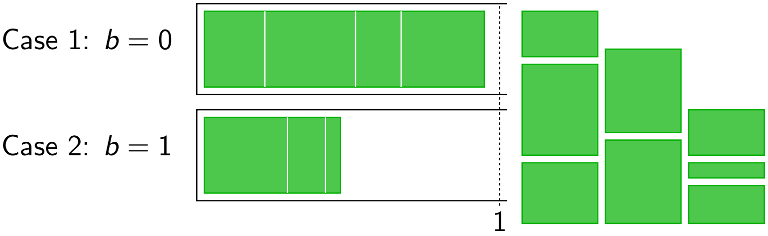
⇨ 4-competitive in expt. and this is tight, but...

Theorem

There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`



Guess one bit and act as with advice

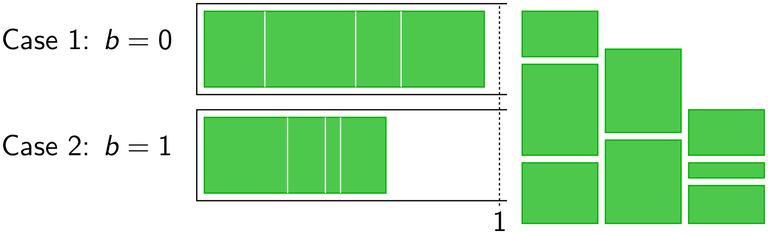
- ⇨ 4-competitive in expt. and this is tight, but...

Theorem

There is a barely random algorithm RAND *for the SKP that uses one random bit and that is* 2*-competitive in expt.*

If b = 0: `act greedily`

If b = 1: `simulate greedy, start when greedy is done`



Suppose, all objects fit into the knapsack. . .

Suppose, all objects fit into the knapsack...

⇨ Greedy strategy optimal, second strategy gains nothing, so

$$\mathbb{E}[\text{comp}(\text{RAND}(I))] = \frac{\text{cost}(\text{OPT})}{\frac{1}{2} \cdot \text{cost}(\text{OPT}) + \frac{1}{2} \cdot 0} = 2$$

Suppose, all objects fit into the knapsack...

⇨ Greedy strategy optimal, second strategy gains nothing, so

$$\mathbb{E}[\text{comp}(\text{RAND}(I))] = \frac{\text{cost}(\text{OPT})}{\frac{1}{2} \cdot \text{cost}(\text{OPT}) + \frac{1}{2} \cdot 0} = 2$$

Suppose, they do not all fit...

Suppose, all objects fit into the knapsack...

⇨ Greedy strategy optimal, second strategy gains nothing, so

$$\mathbb{E}[\text{comp}(\text{RAND}(I))] = \frac{\text{cost}(\text{OPT})}{\frac{1}{2}\cdot\text{cost}(\text{OPT}) + \frac{1}{2}\cdot 0} = 2$$

Suppose, they do not all fit...

⇨ Gains x and y of both strategies are, in the sum, ≥ 1, so

$$\mathbb{E}[\text{comp}(\text{RAND}(I))] = \frac{\text{cost}(\text{OPT})}{\frac{1}{2}\cdot x + \frac{1}{2}\cdot y} \leq \frac{1}{\frac{1}{2}\cdot(x+y)} \leq 2$$

Suppose, all objects fit into the knapsack...

⇨ Greedy strategy optimal, second strategy gains nothing, so

$$\mathbb{E}[\mathrm{comp}(\mathrm{RAND}(I))] = \frac{\mathrm{cost}(\mathrm{OPT})}{\frac{1}{2}\cdot\mathrm{cost}(\mathrm{OPT}) + \frac{1}{2}\cdot 0} = 2$$

Suppose, they do not all fit...

⇨ Gains x and y of both strategies are, in the sum, ≥ 1, so

$$\mathbb{E}[\mathrm{comp}(\mathrm{RAND}(I))] = \frac{\mathrm{cost}(\mathrm{OPT})}{\frac{1}{2}\cdot x + \frac{1}{2}\cdot y} \leq \frac{1}{\frac{1}{2}\cdot(x+y)} \leq 2$$

Theorem

This is the best you can do for the SKP in randomized online computation.

Conclusion

Simple Knapsack Problem

- 1 advice bit suffices to be 2-competitive; surprisingly...
- any additional bit does not help until logarithmic advice

⇨ $(1+\varepsilon)$-competitive algorithm, $\varepsilon > 0$

Simple Knapsack Problem

- 1 advice bit suffices to be 2-competitive; surprisingly. . .
- any additional bit does not help until logarithmic advice
- ⇨ $(1+\varepsilon)$-competitive algorithm, $\varepsilon > 0$
- linear number necessary / sufficient to be optimal
- ⇨ exponential jump to be "a little better"

Simple Knapsack Problem

- 1 advice bit suffices to be 2-competitive; surprisingly...
- any additional bit does not help until logarithmic advice
- ⇨ $(1+\varepsilon)$-competitive algorithm, $\varepsilon > 0$
- linear number necessary / sufficient to be optimal
- ⇨ exponential jump to be "a little better"
- One random bit as powerful as one advice bit
- More random bits do not help

Simple Knapsack Problem

- 1 advice bit suffices to be 2-competitive; surprisingly...
- any additional bit does not help until logarithmic advice
- ⇨ $(1+\varepsilon)$-competitive algorithm, $\varepsilon > 0$
- linear number necessary / sufficient to be optimal
- ⇨ exponential jump to be "a little better"
- One random bit as powerful as one advice bit
- More random bits do not help
- Resource augmentation: Very good (depending on δ) with constant number of advice bits

Simple Knapsack Problem

- 1 advice bit suffices to be 2-competitive; surprisingly...
- any additional bit does not help until logarithmic advice
- ⇨ $(1+\varepsilon)$-competitive algorithm, $\varepsilon > 0$
- linear number necessary / sufficient to be optimal
- ⇨ exponential jump to be "a little better"
- One random bit as powerful as one advice bit
- More random bits do not help
- Resource augmentation: Very good (depending on δ) with constant number of advice bits

General Knapsack Problem

- Not competitive with sub-logarithmic advice
- $(1+\varepsilon)$-competitive with logarithmic advice, $\varepsilon > 0$
- Randomization does not help

Thank you for your attention!