REMOVABLE KNAPSACK AND ADVICE

BÖCKENHAUER, FREI AND ROSSMANITH, 2014

Hoang Long Nguyen

CONTENTS

	Remove	Advice
Offline	X	X
Online	X	X
Online	X	O
Online	O	X
Online	O	O

Discussion and Further Questions

OFFLINE PROBLEM

Generally, we have a set of instances and a set of solutions

Depends on the problem the solution could either be minimize the cost or maximize the gain

The solution is made given that the entire instance known in advance

We interested in the running time of the algorithm to provide solution

Definition 1.1 (Optimization Problem). An *optimization problem* Π consists of a set of *instances* $\mathcal{I}$, a set of *solutions* $\mathcal{O}$, and three functions $\text{sol}\colon \mathcal{I} \to \mathcal{P}(\mathcal{O})$, $\text{quality}\colon \mathcal{I} \times \mathcal{O} \to \mathbb{R}$, and $\text{goal} \in \{\min, \max\}$. For every instance $I \in \mathcal{I}$, $\text{sol}(I) \subseteq \mathcal{O}$ denotes the set of *feasible solutions* for I. For every instance $I \in \mathcal{I}$ and every feasible solution $O \in \text{sol}(I)$, $\text{quality}(I, O)$ denotes the *measure* of I and O. An *optimal solution* for an instance $I \in \mathcal{I}$ of Π is a solution $\textsc{Opt}(I) \in \text{sol}(I)$ such that

$$\text{quality}(I, \textsc{Opt}(I)) = \text{goal}\{\text{quality}(I, O) \mid O \in \text{sol}(I)\} .$$

If $\text{goal} = \min$, we call Π a *minimization problem* and write "cost" instead of "quality." Conversely, if $\text{goal} = \max$, we say that Π is a *maximization problem* and write "gain" instead of "quality."

ONLINE PROBLEM

Generally, the solution is computed given incomplete information

Given this settings, we want an algorithm that:

- Makes decisions given that the input is revealed one by one
- Cannot change any decision that already made

We interested in how much we can achieve when not knowing the whole input in advance

Example: Ski Rental Problem, Job Application, Asset Allocation

Definition 1.4 (Online Problem). An *online problem* Π consists of a set of instances $\mathcal{I}$, a set of *solutions* $\mathcal{O}$, and three functions sol, quality, and goal with the same meaning as for general optimization problems according to Definition 1.1. Every instance $I \in \mathcal{I}$ is a sequence of *requests* $I = (x_1, x_2, \ldots, x_n)$ and every output $O \in \mathcal{O}$ is a sequence of *answers* $O = (y_1, y_2, \ldots, y_n)$, where $n \in \mathbb{N}^+$ (thus, all instances and solutions are finite). An *optimal solution* for an instance $I \in \mathcal{I}$ of Π is a solution $\textsc{Opt}(I) \in \text{sol}(I)$ such that

$$\text{quality}(I, \textsc{Opt}(I)) = \text{goal}\{\text{quality}(I, O) \mid O \in \text{sol}(I)\} .$$

If goal = min, we call Π an *online minimization problem* and write "cost" instead of "quality." Conversely, if goal = max, we say that Π is an *online maximization problem* and write "gain" instead of "quality."

Definition 1.5 (Online Algorithm). Let Π be an online problem and let $I = (x_1, x_2, \ldots, x_n)$ be an instance of Π. An *online algorithm* $\textsc{Alg}$ for Π computes the output $\textsc{Alg}(I) = (y_1, y_2, \ldots, y_n)$, where y_i only depends on $x_1, x_2, \ldots, x_i$ and $y_1, y_2, \ldots, y_{i-1}$; $\textsc{Alg}(I)$ is a feasible solution for I, that is, $\textsc{Alg}(I) \in \text{sol}(I)$.

BOUNDARIES

Upper bound

- The worst-case guarantee that any pre-specified algorithm can achieve in either offline or online settings

Lower bound

- The best any algorithm can achieve in either offline or online problems
- We often use adversary in the proof of lower bound

Generally, in the context of competitive analysis for algorithms,

LB = UB - ϵ

ϵ is a small positive number that is used to indicate a slight improvement to illustrate non-tight bound

OFFLINE SIMPLE KNAPSACK

An example of Offline Problem

Since this is an NP-hard problem, we can't hope for an algorithm to solve for optimization within polynomial time

The **approximation ratio**: what we can achieve for an NP-hard problem when computing in polynomial running time

Intuition: a 2-approximation algorithm means that in the algorithm is never more than twice as bad as the optimal solution

Let's see an example of 2-approximation algorithm that works in polynomial time (KNGREEDY)

Definition 1.3 (Simple Knapsack Problem). The simple knapsack problem is a maximization problem. An instance I is given by a sequence of $n+1$ positive integers $B, w_1, w_2, \ldots, w_n$, where we consider w_i with $1 \leq i \leq n$ to be the weight of the ith object; B is the capacity of the knapsack. A feasible solution for I is any set $O \subseteq \{1, 2, \ldots, n\}$ such that

$$\sum_{i \in O} w_i \leq B \ .$$

The gain of a solution O and a corresponding instance I is given by

$$\text{gain}(I, O) = \sum_{i \in O} w_i \ .$$

The goal is to maximize this number.

2-APPROXIMATION

First sort the objects in the descending order w.r.t their weights (greedy strategy)

Case 1: If $\sum w_i < B$ → KNGREEDY is optimal

Case 2: If $\sum w_i > B$, then:

- Case 2.1: Suppose the highest weight is at least $\frac{B}{2}$ → Pack this object → 2-approximation
- Case 2.2: Suppose all objects have a weight of less than $\frac{B}{2}$ → We keep packing until j^{th} object that can't be packed to the knapsack ($w_j < \frac{B}{2}$). → The space occupied is larger than $\frac{B}{2}$ → 2-approximation

SIMPLE ONLINE KNAPSACK

- How do assess the performance of online algorithm?
- We introduce **competitive ratio**, which has small difference to approximation ratio
  - Now, we are interested in what we can achieve without knowing the entire instance
  - So, we compare the online solution to the hypothetical offline solution
- Both ratios are worst-case measurements
- Illustrated by the following theorem

Definition 6.1 (Online Knapsack Problem). The *online knapsack problem* is an online maximization problem. An instance $I = ((w_1, v_1), (w_2, v_2), \ldots, (w_n, v_n))$ consists of a sequence of n pairs, where we call $w_i \leq 1$ the *weight* and v_i the *value* of the ith object, where $1 \leq i \leq n$. A feasible solution is a set of indices $O \subseteq \{1, 2, \ldots, n\}$ such that

$$\sum_{i \in O} w_i \leq 1 \, .$$

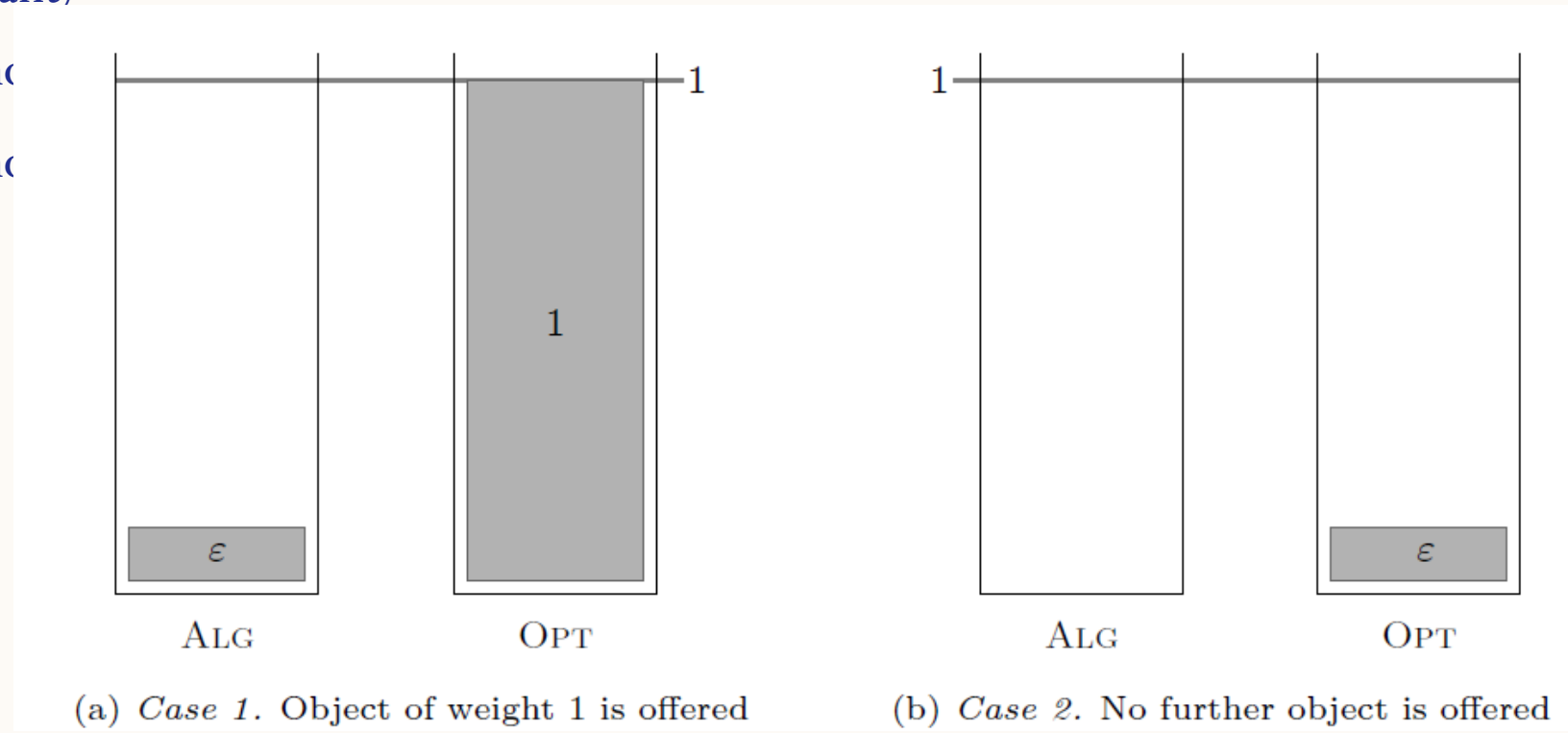
The objective is to choose O such that

$$\sum_{i \in O} v_i$$

is maximized. The objects are offered to an online algorithm one after another, one in every time step. In the corresponding time step, the algorithm must decide whether to pack the object into the knapsack or not. This decision cannot be changed afterwards.

Theorem 6.1. *Let $\varepsilon > 0$. No deterministic online algorithm for the simple knapsack problem is better than strictly $1/\varepsilon$-competitive.*

- In another word, we can't establish a **certain** worst-case solution for the Simple Online Knapsack Problem (because ϵ is **random**).
- Construct an adversary that first offers an object w with a weight of $\epsilon, s.t. \epsilon > 0$
- Then how can we get a certain worst-case performance of any online algorithm? (n-competitive where n is a constant)
- What if nc
- What if nc in the instance?



(a) *Case 1.* Object of weight 1 is offered

(b) *Case 2.* No further object is offered

CHECKPOINT

	Base	Remove	Advice
Offline	UB: 2-approximation LB: 1-approximation	X	X
Online	No bounds	X	X
Online	-	X	O
Online	-	O	X
Online	-	O	O

REMOVABLE KNAPSACK

- A variant of online knapsack problem
- Settings:
  - Allowing the removal of any packed items from the knapsack at any point
  - Possibly once for each item
  - Repacking is not possible

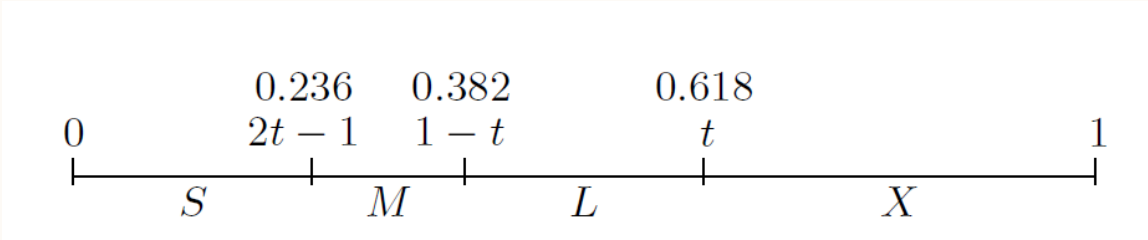
- **Theorem**: there exist a removable online algorithm that achieves ~1.618-competitive ratio for online knapsack problem

PROOF

$Let\ t = \frac{\sqrt{5}-1}{2} \approx 0.618$ be one root of equation $\frac{x}{1-x} = \frac{1}{x}$ and let $\alpha = \frac{1}{t} \approx 1.618$

Theorem: There exist an $\alpha - competitive$ online algorithm for Removable Online KP

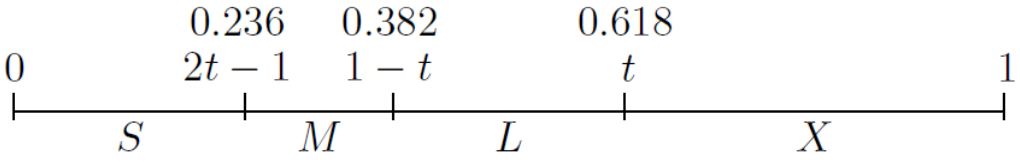
We classified the weight of item into different classes:



Suppose we keep packing item up until i^{th} round. Let Q denote total weight of objects in the knapsack up to this round, u is the weight of the object offered in this round.

Theorem proved if we can prove in the case that $CR < \alpha\ then\ CR' < \alpha$. → competitive ratio at any time step would be smaller than α.

We have possible actions on the next slide



If $Q > t$ -> discard the object (1)

If $u + Q \leq 1$ -> pack the object (2)

If $Q \leq t$ and $u + Q > 1$ => $u > 1 - t$
i.e., object is either L or XL size
- If $u \in X$ -> pack the object & discard others (3)

- If $u \in L$:
  - If the KP includes only S & M:
    -> Pack object and discard other objects one by one until $Q \leq 1$ (4)

  - If the KP includes only one L and (none or S):
    - If $u + L \leq 1$ ~~-> KP contains u & L~~
      -> Make KP contains only these two objects (5)

    - If $u < L$
      -> Hold object and discard L

    - Else discard object

Denoted CR the competitive ratio begore round i

$$CR = \frac{OPT(I)}{Q}$$

Denoted CR' the competitive ratio after round i:

$$CR' = \frac{OPT(I)'}{Q'}$$

At $T = 1$; $CR = 1 < \alpha$

If (1) & (3) occur:

=> $Q' > t$; Since $OPT(I)' \leq 1$

=> $CR' = \frac{OPT(I)'}{Q'} < \frac{1}{t} = \alpha$

If (2) occur:

$$\frac{OPT(I)'}{Q'} < \frac{OPT(I) + u}{Q + u} < \frac{OPT(I)}{Q} < \alpha$$

If (4) occur:

Let the current items in KP $\{b_1, b_2, \ldots, b_M\}$ (discard order)

=> Let b_i the last object discarded

=> $b_i + b_{i+1} + b_{i+2} + \ldots + b_M + u > 1$

and

$Q' = b_{i+1} + b_{i+2} + \ldots + b_M + u \leq 1$

Since the weight of b_i at most $1 - t$

=> $Q' > 1 - w_{b_i} \geq t$

If (5) occur: => $u > 1 - t$ & $L > 1 - t$

=> $Q' > 2(1 - t) = 3 + \sqrt{5} > t$

Theorem 2: There is no online algorithm for removable online knapsack problem that has CR > $\alpha - \epsilon, for\ any\ \epsilon > 0$.

$Let\ t = \frac{\sqrt{5}-1}{2} \approx 0.618$ again and the adversary gives you two items x_1 and x_2 such that it has weight of:

$t + \epsilon', for\ a\ small\ \epsilon' > 0$ and, $1 - t$, respectively.

Either x_1 or x_2 can stay in the KP since sum of its weight > 1, if:

You pick x_2, the game stops, $CR = \frac{t+\epsilon'}{1-t} > \alpha$

You pick x_1, it gives you $x_3, w_{x_3} = t$

- The OPT = 1 (contains $x_2\ and\ x_3$) → $CR = \frac{1}{t+\epsilon'}$
- $\frac{1}{t+\epsilon'} > \frac{1}{t} - \epsilon = \alpha - \epsilon$

ONLINE KNAPSACK WITH ADVICE

- An oracle which sees the whole input in advance and encode any binary information about this input
- As we just seen, we just arbitrarily bad without advice
- We only consider one bit of advice here
- The Theorem: there exist a 2-competitive online algorithm using one bit of advice

PROOF

Consider an algorithm called KNONE reads one bit of advice at the beginning

Depending on its value KNONE either follows a greedy strategy or waits for an object with a weight of more than 1/2.

Case 1. If such object exists, the first bit of advice tape turns to 1, tells KNONE to wait and pack that object → 2-competitive

Case 2. If there is no such object in the instance, first bit of advice tape turns to 0, KNONE follows greedy strategy, packing every objects until not possible → 2-competitive

Theorem 2: There is no online algorithm for the online knapsack problem with one advice bit that has CR $> 2 - \epsilon, for\ any\ \epsilon > 0$.

REMOVABLE KNAPSACK WITH ADVICE

Theorem: There is a $\sqrt{2}-$ *competitive* online algorithm for removeable online knapsack using one bit of advice.

Let's describe a simple $\frac{3}{2}-$ *competitive* one advice bit algorithm

A single advice bit will indicate if the OPT contains more than one item in the interval $[\frac{1}{3}, \frac{2}{3}]$.

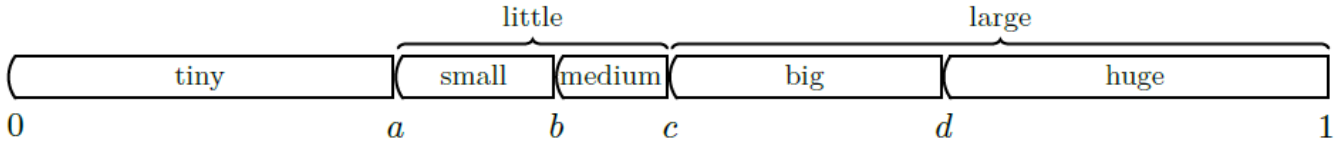
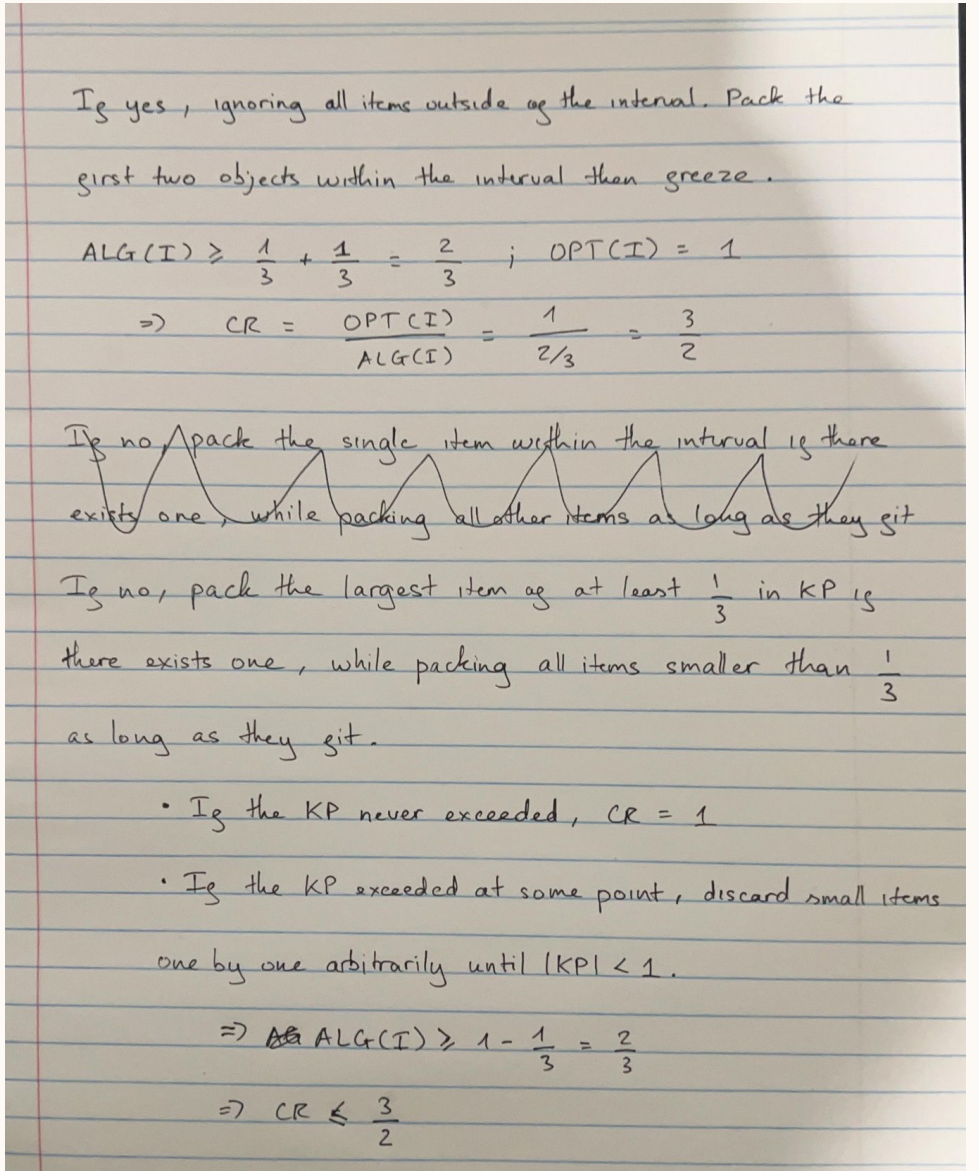


Figure 2 The partition of the interval $(0, 1]$ of possible sizes into the five subintervals used in the proof of Theorem 6—namely $(0, a]$, $(a, b]$, $(b, c]$, $(c, d]$, and $(d, 1]$—plus the corresponding class names. The values are $a = 1 - 1/\sqrt{2} \approx 0.293$, and $b = \sqrt{2} - 1 \approx 0.414$, and $c = 1/2$, and $d = 1/\sqrt{2} \approx 0.707$.



If yes, ignoring all items outside of the interval. Pack the first two objects within the interval then freeze.

$ALG(I) \geq \frac{1}{3} + \frac{1}{3} = \frac{2}{3}$; $OPT(I) = 1$

$\Rightarrow CR = \frac{OPT(I)}{ALG(I)} = \frac{1}{2/3} = \frac{3}{2}$

If no, pack the single item within the interval if there exists one, while packing all other items as long as they fit

If no, pack the largest item of at least $\frac{1}{3}$ in KP if there exists one, while packing all items smaller than $\frac{1}{3}$ as long as they fit.

- If the KP never exceeded, $CR = 1$
- If the KP exceeded at some point, discard small items one by one arbitrarily until $|KP| < 1$.

$\Rightarrow ALG(I) \geq 1 - \frac{1}{3} = \frac{2}{3}$

$\Rightarrow CR \leq \frac{3}{2}$

REMOVABLE KNAPSACK WITH ADVICE

Theorem 2: No algorithm for removeable online knapsack using one bit of advice is better than $\frac{1+\sqrt{17}}{4} \sim 1.2808 - competitive$

Proof:

Define $\psi = \frac{4}{1+\sqrt{17}}$

Because any two of these items sum up to over 1, the advice algorithm can keep at most one of them in the knapsack

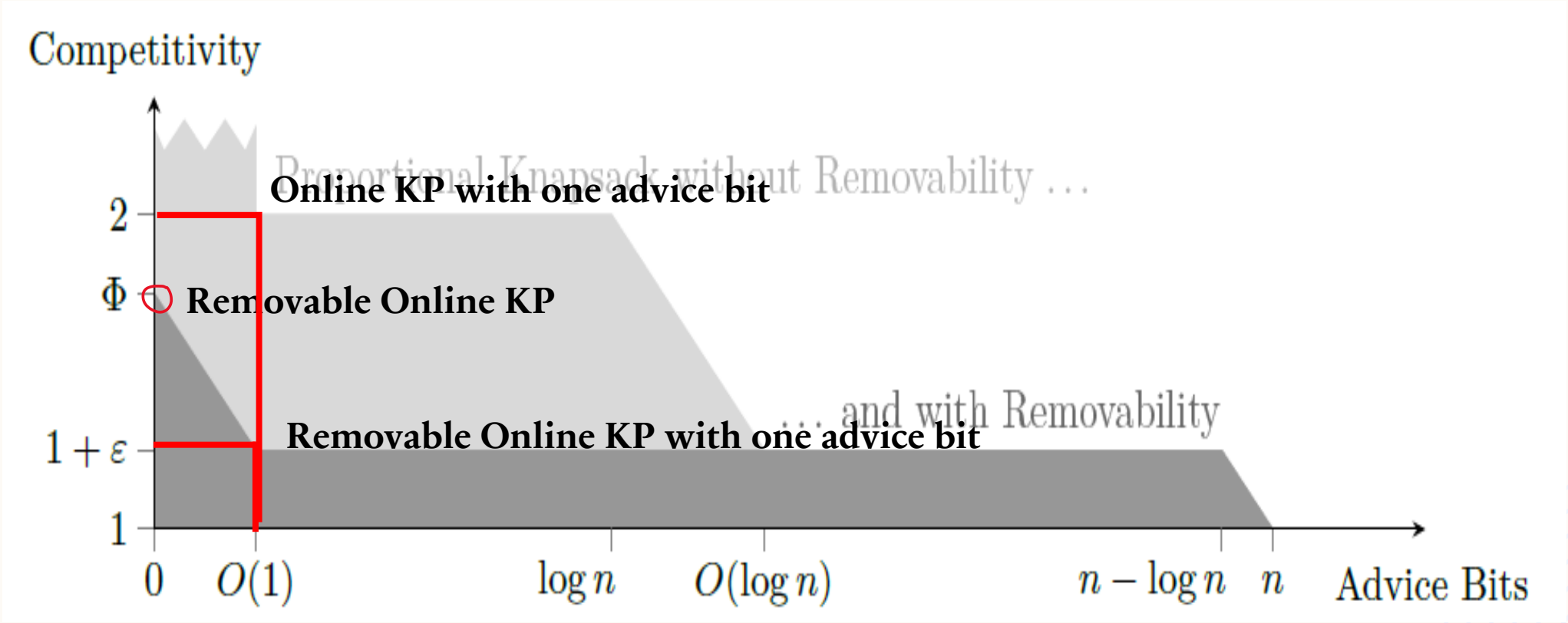
One bit of advice is not enough to make a perfect decision for three separate instances because the bit can only distinguish between two courses of action.

	x_1	x_2	x_3	y_2	y_3	optimal	second best	ratio
I_1:	ψ	ψ^2	$1-\psi^2+\varepsilon$			ψ	ψ^2	ψ/ψ^2
I_2:	ψ	ψ^2	$1-\psi^2+\varepsilon$	$1-\psi^2$		1	$2(1-\psi^2)+\varepsilon$	$1/(2(1-\psi^2)+\varepsilon)$
I_3:	ψ	ψ^2	$1-\psi^2+\varepsilon$		$\psi^2-\varepsilon$	1	ψ	$1/\psi$

Implies sub-optimal CR for at least one instance

The competitive ratio can't be better than the minimum of the last column which is $\frac{1}{\psi} \sim \frac{(1+\sqrt{17})}{4}$

SUMMARY


Competitivity
Proportional Knapsack without Removability . . .
Online KP with one advice bit
2
Φ
Removable Online KP
. . . and with Removability
Removable Online KP with one advice bit
1 + ε
1
0
O(1)
log n
O(log n)
n − log n
n
Advice Bits

SUMMARY

	Boundaries	Remove	Advice
Offline		X	X
Online		X	X
Online		X	O
Online		O	X
Online		O	O

QUESTIONS?

THANK YOU