Online bin packing with advice of small size

Spyros Angelopoulos

Christoph Dürr

Sorbonne Universités, UPMC Paris 06

Shahin Kamali

Waterloo

Marc Renault

Adi Rosén

Université Paris Diderot

June 2015 Eindhoven

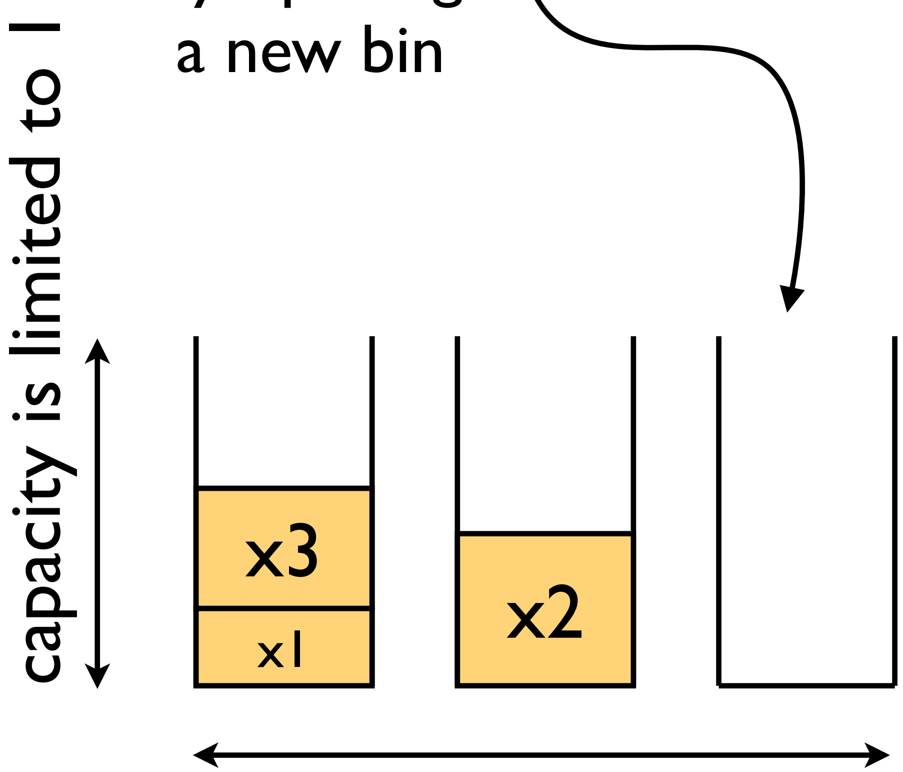
Outline

- The model
- An upper bound
- A lower bound

Online Bin Packing

input: x_1, x_2, x_3, x_4,?,?,. . .

decide where to pack item, possibly by opening a new bin



Asymptotic competitve ratio R

$\forall \sigma : A(\sigma) \leq R \cdot OPT(\sigma) + \text{constant}$

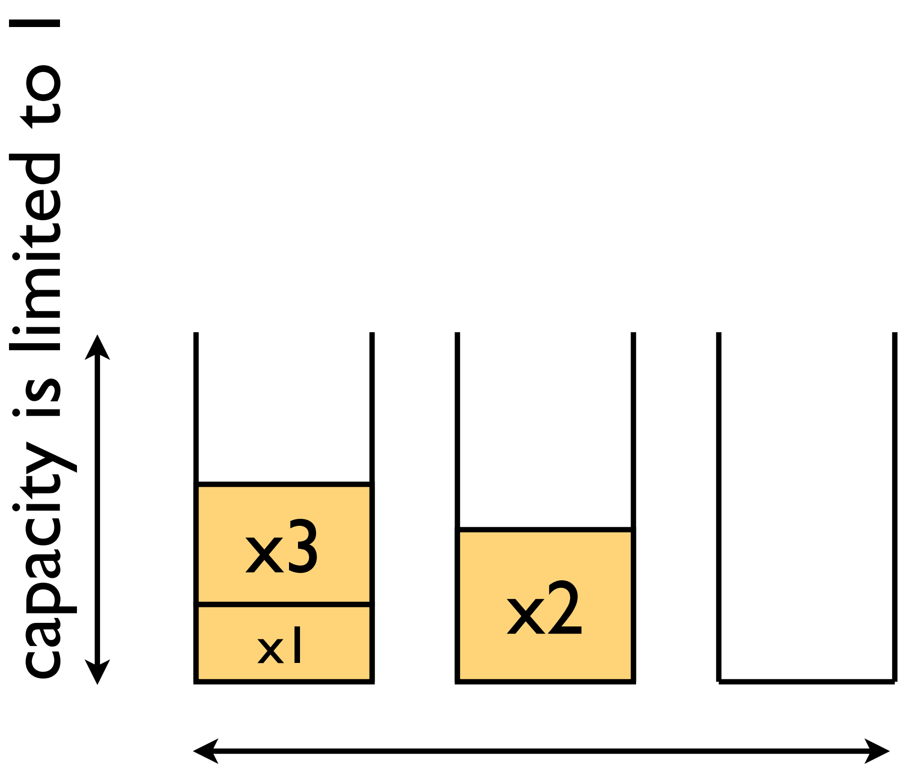
instance

bins used by A

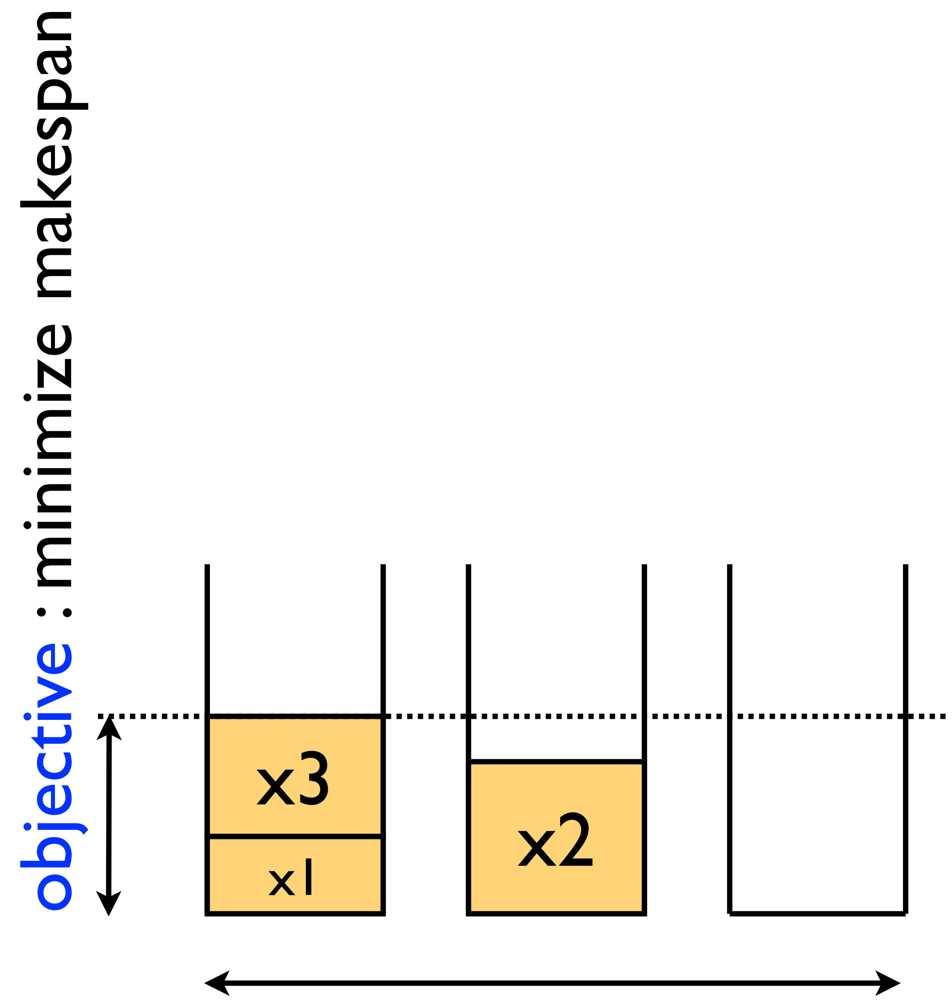
independent of σ

objective : minimize number of opened bins

Bin Packing is orthogonal to scheduling



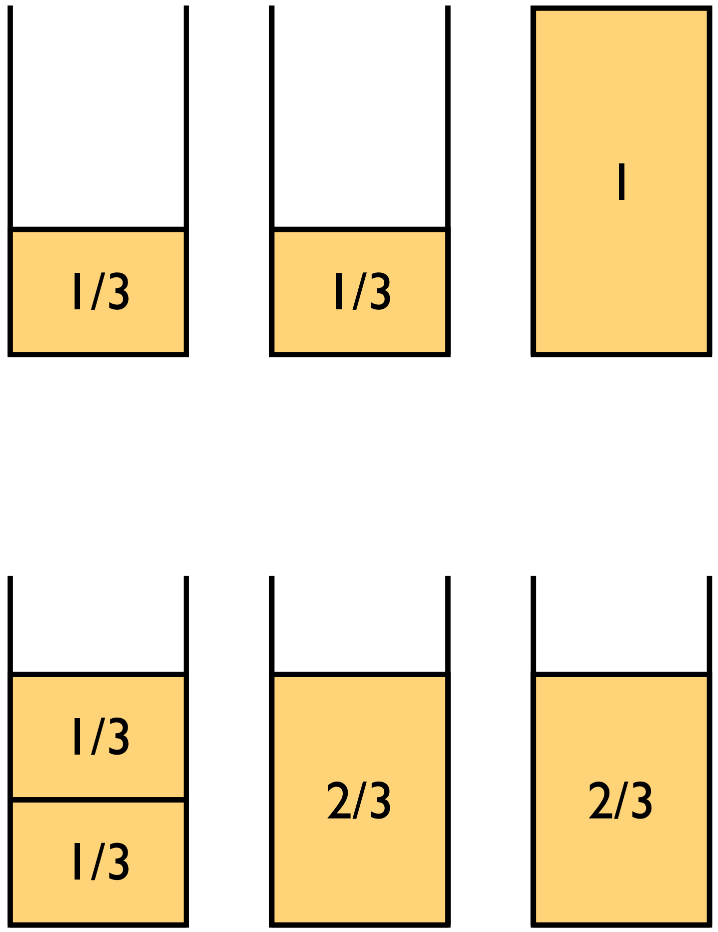
objective : minimize number of opened bins



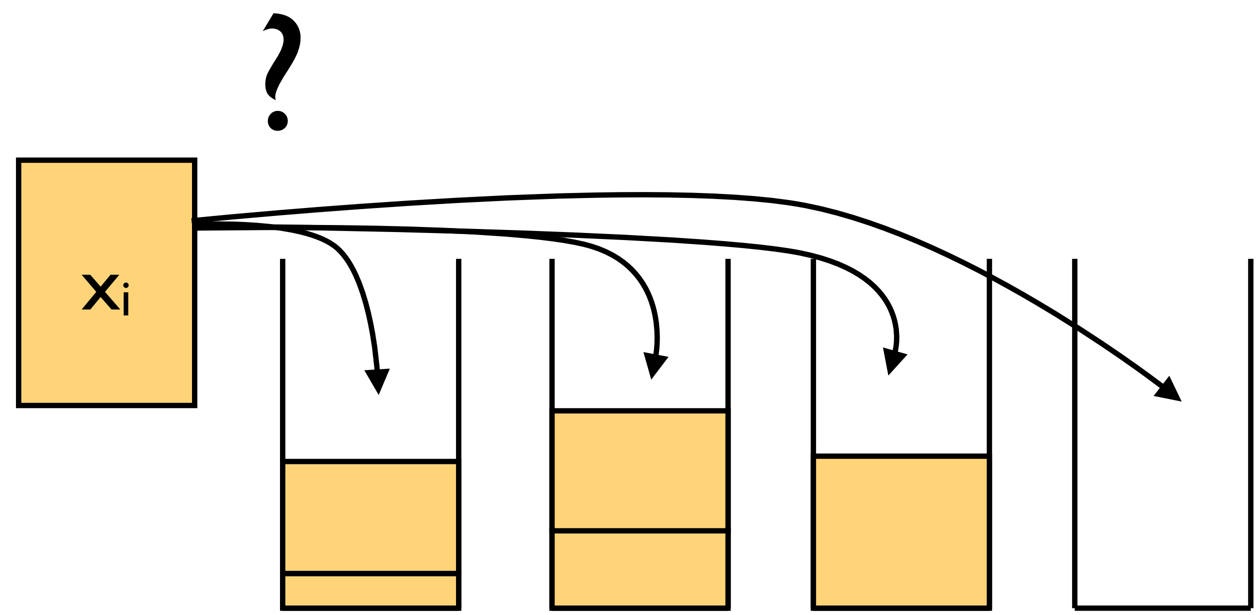
fixed number of identical machines

Example

Algorithm gets two 1/3 items. How to pack them?
Each choice is bad in some scenario.



Some known results



competitive ratio

FirstFit: place each item in first bin where item fits (order in opening)

1.7

1.588 Harmonic++

1.54037

Lower bound holds even for randomized online algorithms

1

Advice model

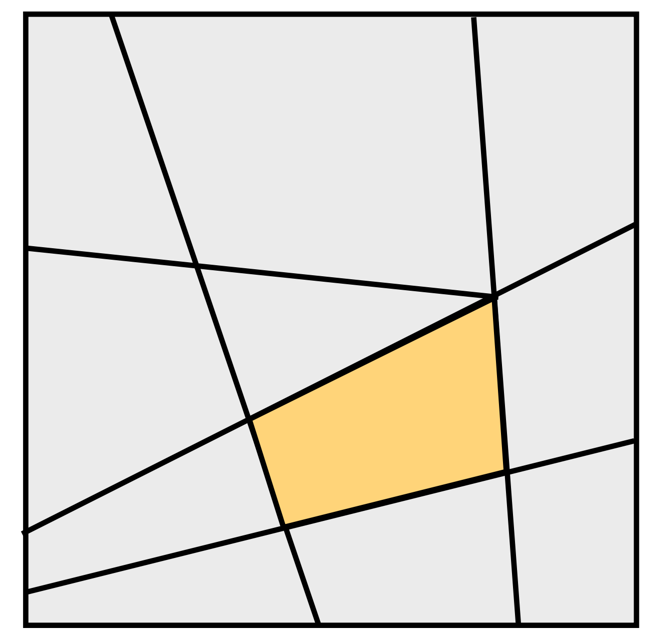
input: $x_1, x_2, x_3, x_4, ?, ?, \ldots$ online problem

intermediate problems:
lookahead, etc.

- Algorithm gets an advice string, which is function of the input x
- Advice function and algorithm are designed together

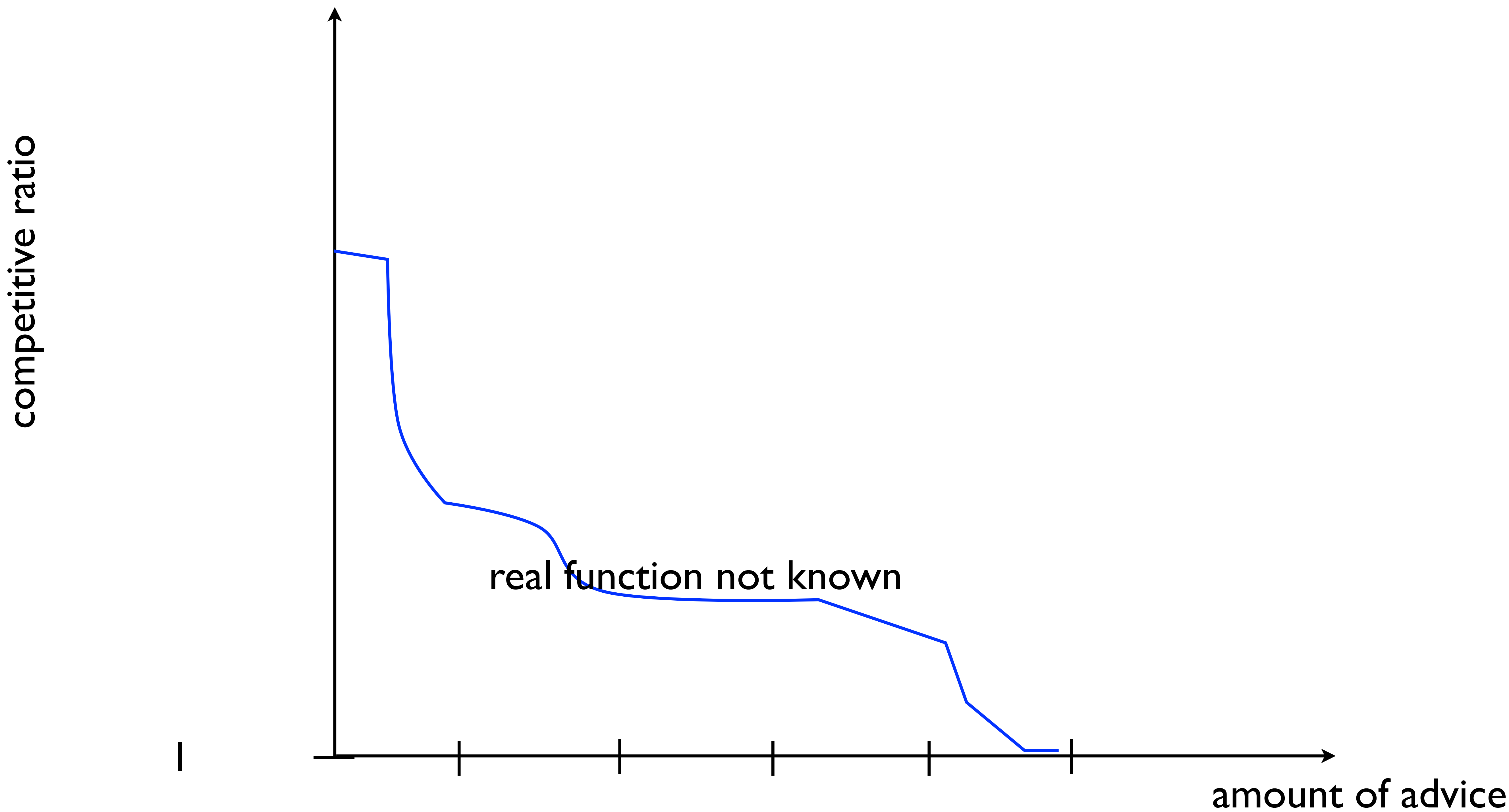
input: $x_1, x_2, x_3, x_4, x_5, \ldots, x_n$ offline problem

set of all instances

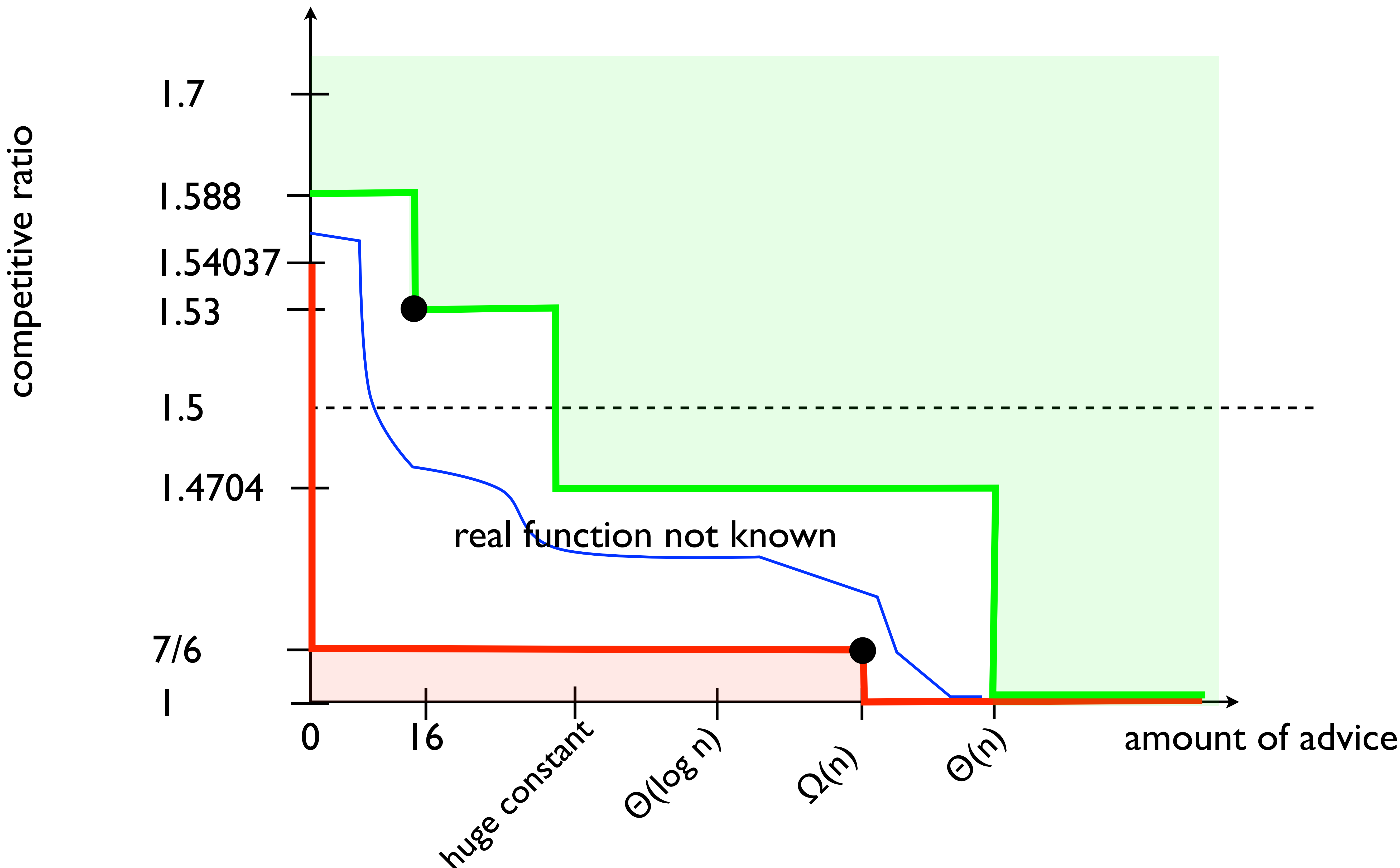


algorithm knows by advice that instance has some properties and can exploit them

Competitive ratio depends on amount of advice



Competitive ratio depends on amount of advice

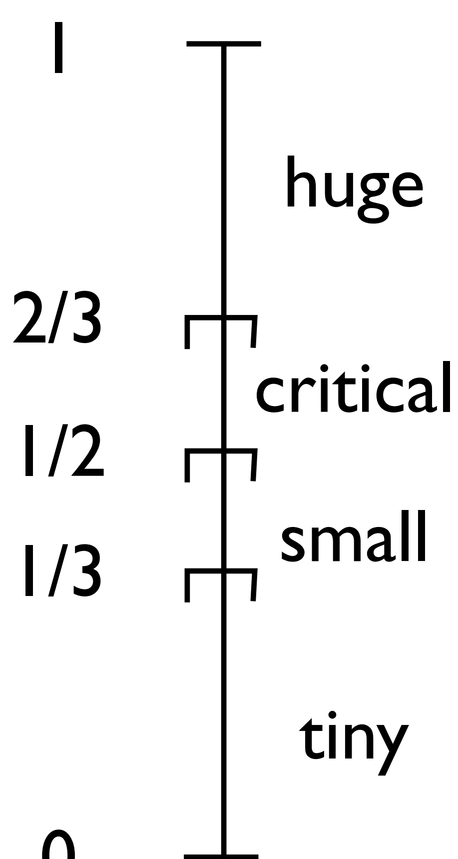


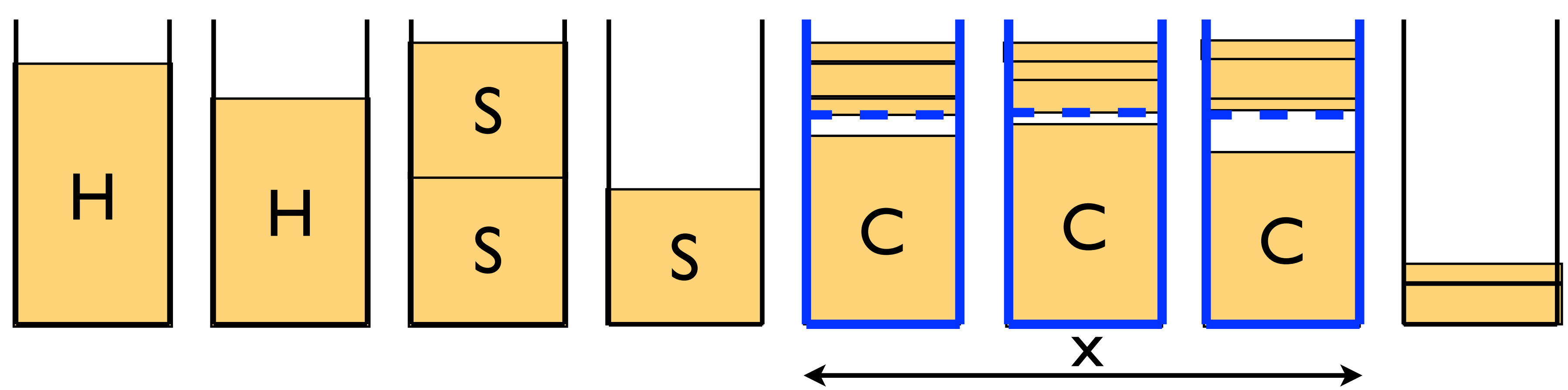
Outline

- The model
- An upper bound
- A lower bound

Classify items

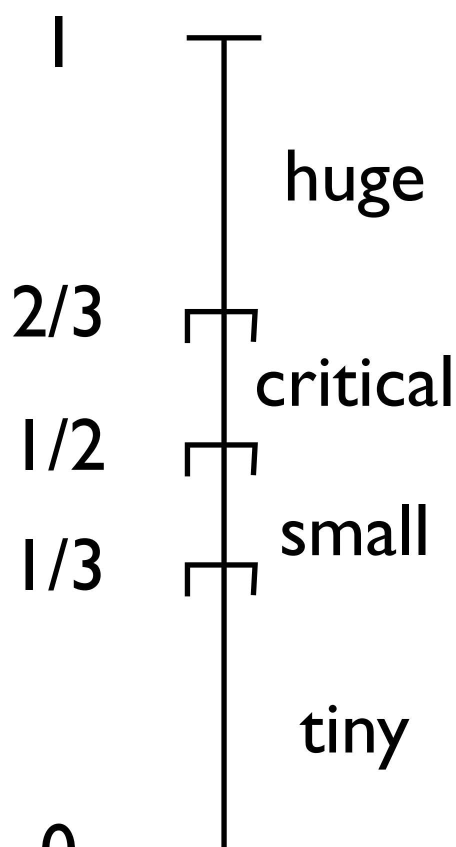
- Algorithm ReserveCritical
- has ratio 1.5, needs Θ(log n) bits of advice telling x, the number of critical items
- place huge items alone in dedicated bins
- place small items by pairs in dedicated bins
- open x bins with 2/3 reserved for critical items
- place critical items in their reserved bins
- place tiny items using FirstFit in remaining space

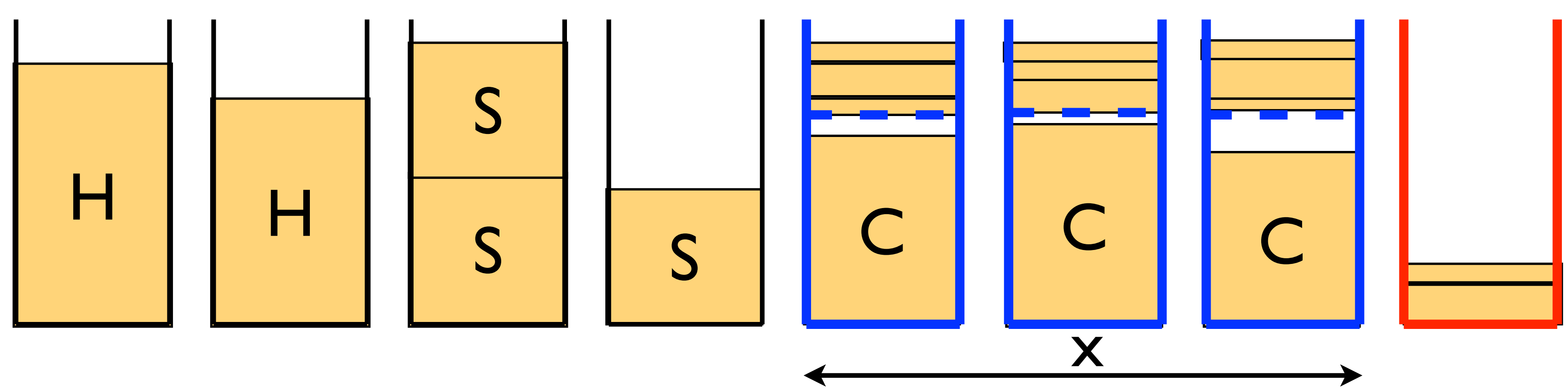




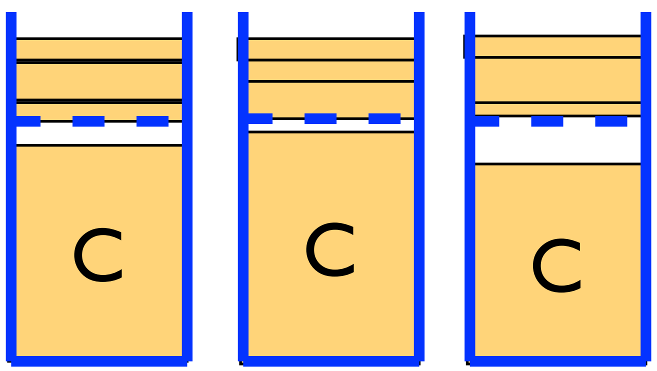
Approximate x

- Algorithm RedBlue
- Let y be number of bins opened for tiny items by ReserveCritical
- Advice : $\beta = i/2^k$ — for $0 \le i < 2^k$ and constant k with $\beta \le x/(x+y) < \beta + 1/2^k$
- Algorithm opens a blue bin for each critical item, possibly also a red bin *to ensure right proportion*
- gives ratio $1.5 + 15/2^{k/2+1}$

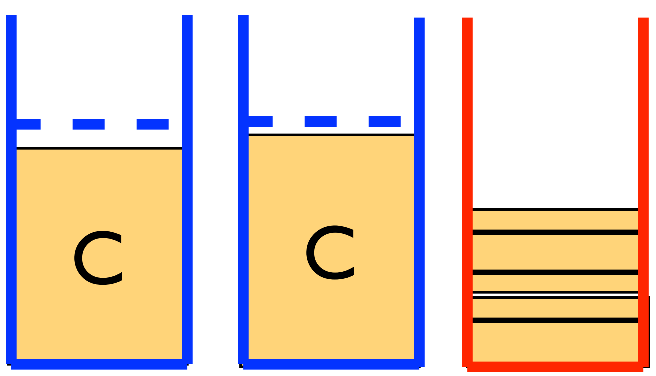




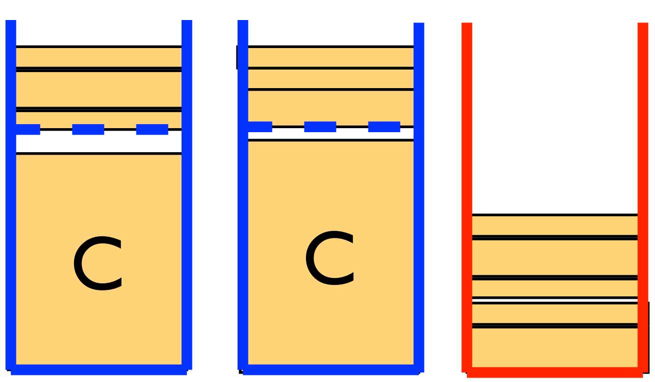
3 cases for RedBlue



- $\beta > 1-1/2^{k/2}$: Every opened bin is blue. Place critical items in their reserved space, and tiny items by FirstFit in their reserved space.
  Ratio $\leq 1.5+7.5/(2^{k/2})$



- $\beta < 1/2^{k/2}$: Place critical items in blue bins, label opened bins as blue. Place tiny items in red bins, label opened bins as red.
  Ratio $\leq 1.5+3/(2^{k}-2)$

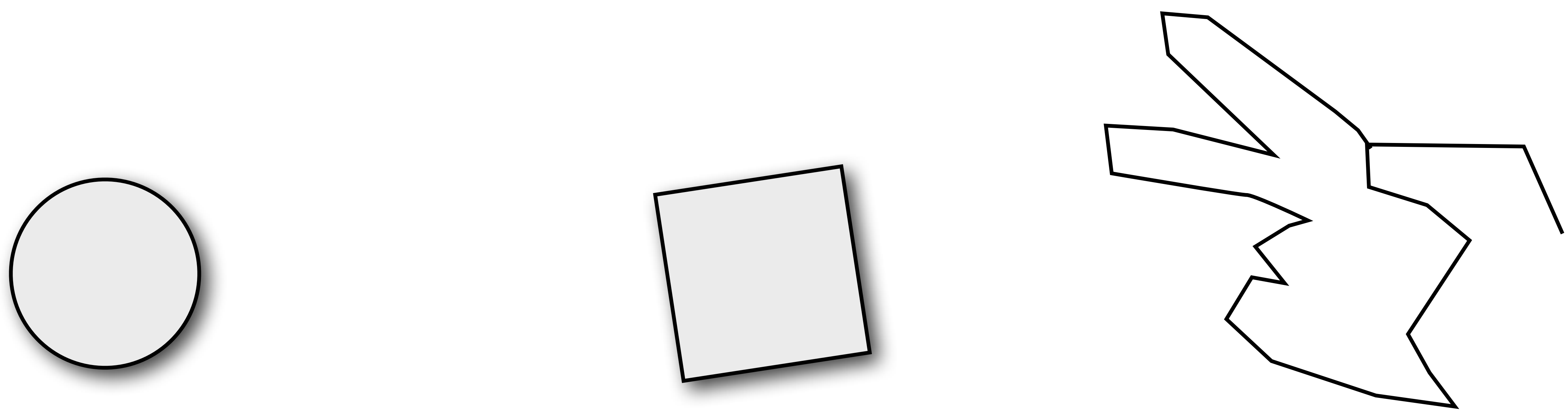


- $1/2^{k/2} \leq \beta \leq 1-1/2^{k/2}$: Place critical items in their reserved space, label opend bins as blue. Place tiny items by FirstFit in blue and red bins. Label opened bins preferably blue if (#blue+1)/(#blue+#red +1)$\leq\beta$, otherwise red.
  Ratio $\leq 1.5+15/(2^{k/2+1})$

Outline

- The model
- An upper bound
- A lower bound

What is more even more stupid than the stone-paper-scissor game?



binary string guessing problem

0	1	1	0	0	1	0	1
0	1	0	1	$\boxed{0}$			
win	win			win			

- Algorithm needs to guess a binary string
- After he announces a bit, he immediatly learns if it was a match or not
- Say we have the promize that the hidden strings has as many 0s than 1s.
- In order to guess correctly at least an α fraction, b(n) advice bits are necessary with

$$b(n) = (1 + (1-\alpha)\log(1-\alpha) + \alpha\log\alpha)n - e(n) - 1$$
$$e(n) = \lceil \log(n/2+1) \rceil + 2\lceil \log\left(\lceil \log(n/2+1)\rceil + 1\right)\rceil + 1$$

reduce to bin packing

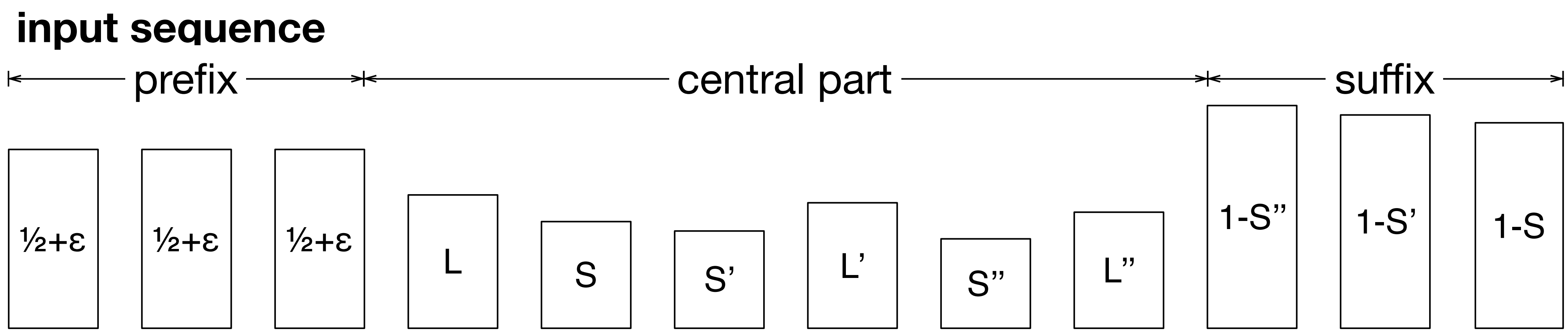
From a

hidden string x to the string guessing problem (as many 0s than 1s)

1 0 0 1 0 1

Construct a

sequence of 2n items. The n central items are smaller than 1/2-ε.
The n/2 largest of them correspond to 1s in x.

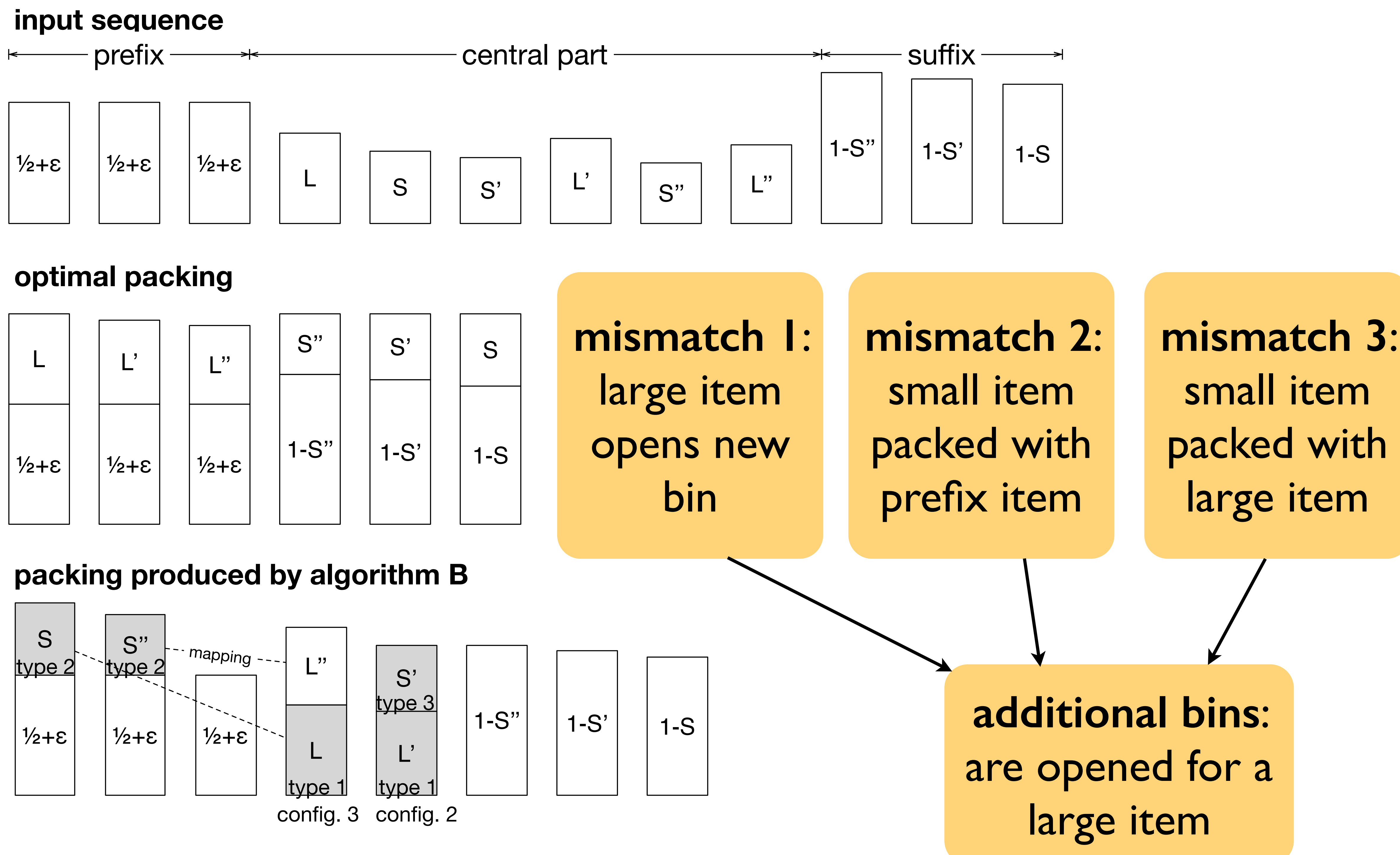


The n/2 suffix items are exact complements to the n/2 smaller central items.

Such that

An algorithm B that would open OPT+k bins,
would correspond to an algorithm A that mismatches at most 3k bits from x.

Algorithm A: run algorithm B, if it opens bin for central item, labeled it 1 otherwise 0



=> If B opens k bins, then A made at most 3k mismatches

Can you give me an advice ?

End the talk.