Online Bin Packing with Advice

Joan Boyar[1], Shahin Kamali[2],
Kim S. Larsen[1], Alejandro López-Ortiz[2]

1 University of Southern Denmark, Denmark
2 University of Waterloo, Canada

July 7, 2014

Overview

1. The bin packing problem: offline and online

2. Advice complexity results for bin packing

3. Open problems

Section 1

The bin packing problem: offline and online

Bin Packing Problem

Input: items of various sizes $\in (0, 1]$

Output: packing of all items into unit size bins

Goal: use minimum number of bins

Bin Packing Problem

Input: items of various sizes $\in (0, 1]$

Output: packing of all items into unit size bins

Goal: use minimum number of bins

Applications: storage, cutting stock...

Offline Bin Packing Problem

The problem is NP-hard; Reduce from 2-PARTITION.

First-Fit-Decreasing has an approximation ratio of $11/9 \approx 1.22$ [Johnson,Demers,Ullman,Garey,Graham, 1974]

There is an asymptotic PTAS for the problem [de la Vega,Lueker, 1981]

Online Bin Packing Problem

Request sequence is revealed in a sequential, online manner.

Examples:

- Next-Fit
- First-Fit
- Best-Fit
- Harmonic, Harmonic++

First-Fit vs. Next-Fit — Online

First-Fit

- Find the first open bin with enough space, and place the item there
- If such a bin does not exist, open a new bin

First-Fit vs. Next-Fit — Online

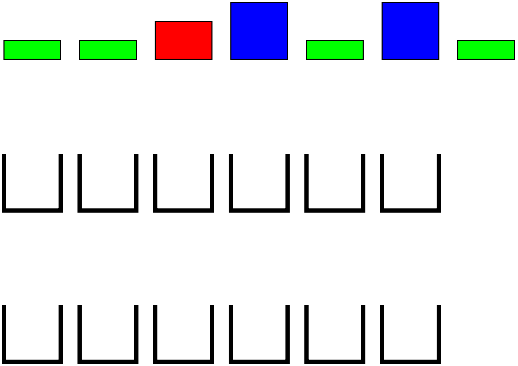
First-Fit

- Find the first open bin with enough space, and place the item there
- If such a bin does not exist, open a new bin

Next-Fit

- Put item in current open bit, if it fits
- Otherwise, close that bin and open a new current bin

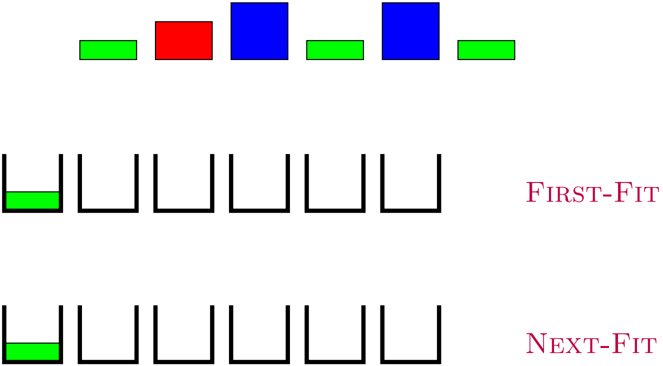
First-Fit vs. Next-Fit — Online



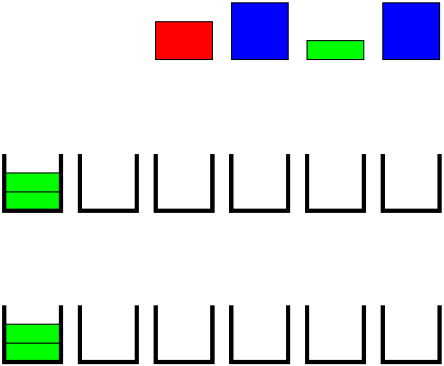
First-Fit

Next-Fit

First-Fit vs. Next-Fit — Online



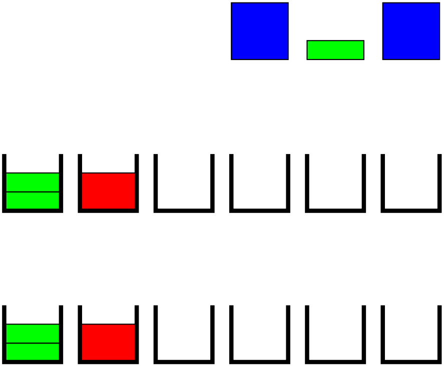
First-Fit vs. Next-Fit — Online



First-Fit

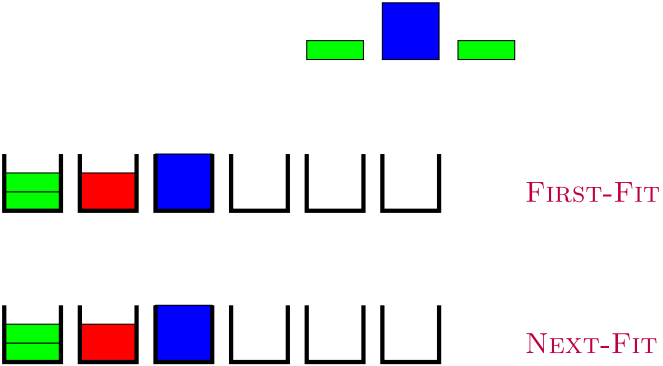
Next-Fit

First-Fit vs. Next-Fit — Online

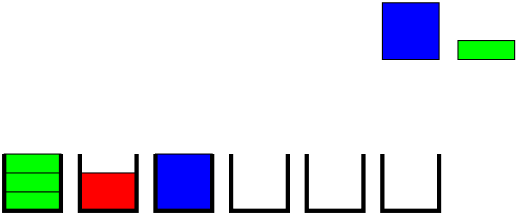


First-Fit

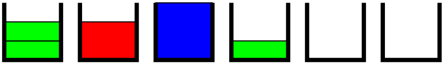
Next-Fit


FIRST-FIT
NEXT-FIT

First-Fit vs. Next-Fit — Online

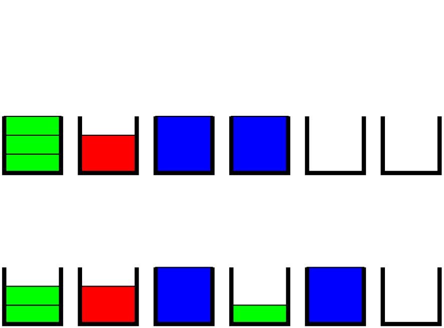


First-Fit



Next-Fit

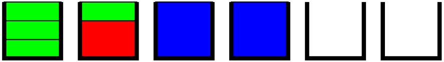
First-Fit vs. Next-Fit — Online



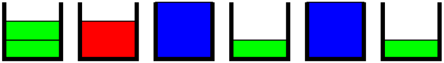
First-Fit

Next-Fit

First-Fit vs. Next-Fit — Online

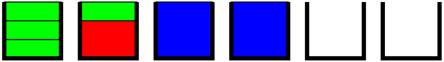


First-Fit

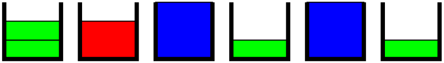


Next-Fit

First-Fit vs. Next-Fit — Online



First-Fit Result: 4



Next-Fit Result: 6

Competitive Analysis

Compare the performance of an online algorithm, ALG, with an optimal **offline** algorithm, OPT:

- OPT knows the whole sequence in the beginning.

Competitive ratio of ALG is the maximum ratio between the cost of ALG and OPT for serving the same sequence.

Competitive Analysis

Next-Fit has competitive ratio 2
[Johnson, 1974]

Best-Fit and First-Fit have competitive ratio 1.7
[Johnson,Demers,Ullman,Garey,Graham, 1974]

Best known online algorithm (Harmonic++) has competitive ratio 1.58889
[Seiden, 2002]

Competitive Analysis

Next-Fit has competitive ratio 2
[Johnson, 1974]

Best-Fit and First-Fit have competitive ratio 1.7
[Johnson,Demers,Ullman,Garey,Graham, 1974]

Best known online algorithm (Harmonic++) has competitive ratio 1.58889
[Seiden, 2002]

No online algorithm has a competitive ratio less than 1.54037
[Balogh,Békési,Galambos, 2012]

Competitive Analysis

Next-Fit has competitive ratio 2
[Johnson, 1974]

Best-Fit and First-Fit have competitive ratio 1.7
[Johnson,Demers,Ullman,Garey,Graham, 1974]

Best known online algorithm (Harmonic++) has competitive ratio 1.58889
[Seiden, 2002]

No online algorithm has a competitive ratio less than 1.54037
[Balogh,Békési,Galambos, 2012]

Recall that offline First-Fit-Decreasing has approximation ratio ≈ 1.22.

- A big gap between quality of online and offline solutions.
- What about an "almost online" algorithm?

Section 2

Advice complexity results for bin packing

Advice Model for Online Bin Packing Problem

Relax “absolutely no knowledge” assumption:

Advice Model for Online Bin Packing Problem

Relax "absolutely no knowledge" assumption:

Same advice model as previous talk
[Böckenhauer,Komm,Královič,Královič,Mömke, 2009]

Algorithms get $b(n)$ bits of advice for sequences of length n:

Advice Model for Online Bin Packing Problem

Relax "absolutely no knowledge" assumption:

Same advice model as previous talk
[Böckenhauer,Komm,Královič,Královič,Mömke, 2009]

Algorithms get $b(n)$ bits of advice for sequences of length n:

The advice is generated by an offline oracle.

Advice Model for Online Bin Packing Problem

Relax “absolutely no knowledge” assumption:

Same advice model as previous talk
[Böckenhauer,Komm,Královič,Královič,Mömke, 2009]

Algorithms get $b(n)$ bits of advice for sequences of length n:

The advice is generated by an offline oracle.

The advice is written on a tape and can be accessed by the online algorithm at any time.

Advice Model for Online Bin Packing Problem

Relax "absolutely no knowledge" assumption:

Same advice model as previous talk
[Böckenhauer,Komm,Královič,Královič,Mömke, 2009]

Algorithms get $b(n)$ bits of advice for sequences of length n:

The advice is generated by an offline oracle.

The advice is written on a tape and can be accessed by the online algorithm at any time.

- There are other advice models for bin packing
  - Original: [Dobrev, Královič, Markou, 2009]
  - Advice with request: [Fraigniaud,Korman,Rosén, 2011]

Relevant Questions

For a sequence of fixed length

- How many bits of advice are required (sufficient) to achieve an optimal solution?
- How many bits of advice are sufficient to outperform all online algorithms?
- How good can the competitive ratio be with advice of linear/sublinear size?

Relevant Questions

For a sequence of fixed length

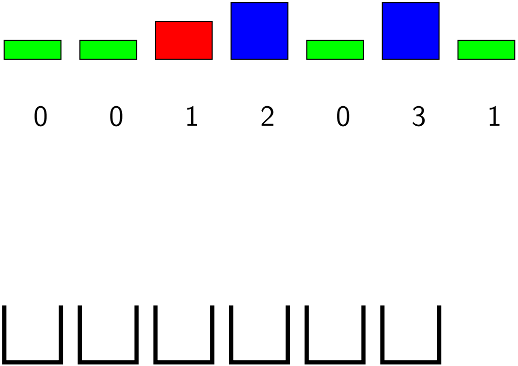
- How many bits of advice are required (sufficient) to achieve an optimal solution?
- How many bits of advice are sufficient to outperform all online algorithms?
- How good can the competitive ratio be with advice of linear/sublinear size?

Is there useful advice one could reasonably get (without knowing $\textsc{Opt}$)?

Optimal Solution with Advice

How many bits of advice are sufficient to achieve an optimal solution?

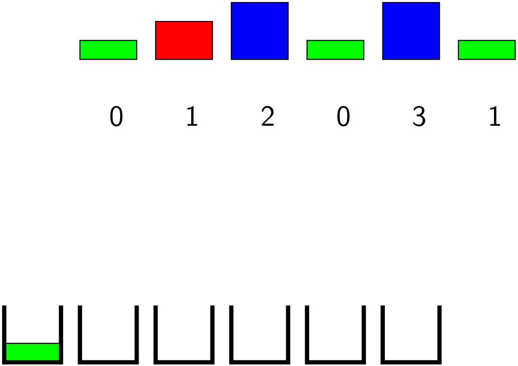
- Advice for each item: index of target bin in OPT's packing.
- $n\lceil \log \mathrm{OPT}(\sigma) \rceil$ bits of advice are sufficient



Optimal Solution with Advice

How many bits of advice are sufficient to achieve an optimal solution?

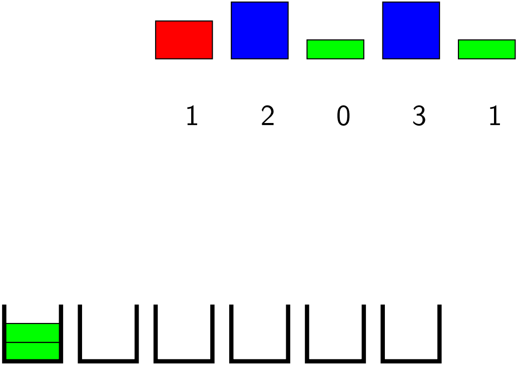
- Advice for each item: index of target bin in $\textsc{Opt}$'s packing.
- $n\lceil \log \textsc{Opt}(\sigma) \rceil$ bits of advice are sufficient



Optimal Solution with Advice

How many bits of advice are sufficient to achieve an optimal solution?

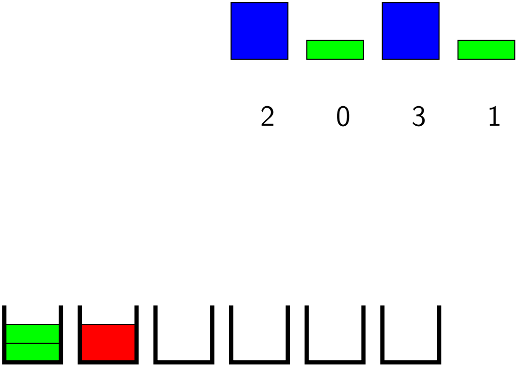
- Advice for each item: index of target bin in $\textsc{Opt}$'s packing.
- $n\lceil \log \textsc{Opt}(\sigma) \rceil$ bits of advice are sufficient



Optimal Solution with Advice

How many bits of advice are sufficient to achieve an optimal solution?

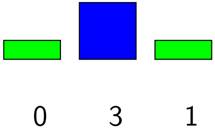
- Advice for each item: index of target bin in $\textsc{Opt}$'s packing.
- $n\lceil\log \textsc{Opt}(\sigma)\rceil$ bits of advice are sufficient

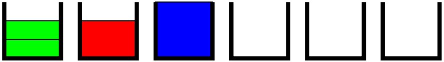


Optimal Solution with Advice

How many bits of advice are sufficient to achieve an optimal solution?

- Advice for each item: index of target bin in $\textsc{Opt}$'s packing.
- $n\lceil \log \textsc{Opt}(\sigma) \rceil$ bits of advice are sufficient

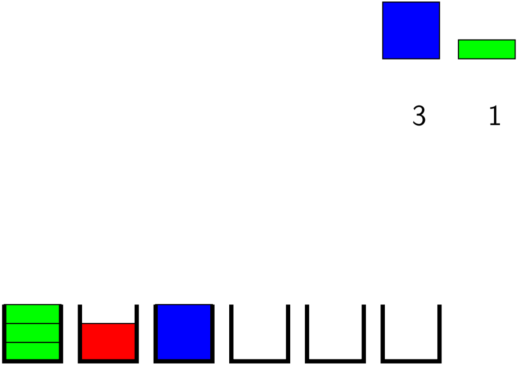




Optimal Solution with Advice

How many bits of advice are sufficient to achieve an optimal solution?

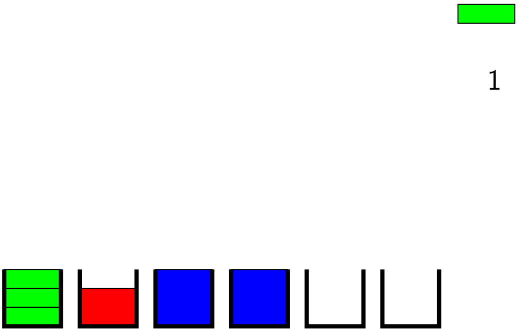
- Advice for each item: index of target bin in $\textsc{Opt}$'s packing.
- $n\lceil \log \textsc{Opt}(\sigma) \rceil$ bits of advice are sufficient



Optimal Solution with Advice

How many bits of advice are sufficient to achieve an optimal solution?

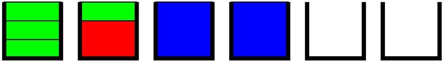
- Advice for each item: index of target bin in $\textsc{Opt}$'s packing.
- $n\lceil \log \textsc{Opt}(\sigma) \rceil$ bits of advice are sufficient



Optimal Solution with Advice

How many bits of advice are sufficient to achieve an optimal solution?

- Advice for each item: index of target bin in $\textsc{Opt}$'s packing.
- $n\lceil \log \textsc{Opt}(\sigma) \rceil$ bits of advice are sufficient



Optimal Solution with Advice

In fact, $(n - 2\,\mathbf{Opt}(\sigma))\log \mathbf{Opt}(\sigma)$ bits of advice are required to achieve an optimal packing.

Optimal Solution with Advice

In fact, $(n - 2\,\mathbf{Opt}(\sigma))\log\mathbf{Opt}(\sigma)$ bits of advice are required to achieve an optimal packing.

Comparison:
$n\lceil\log\mathbf{Opt}(\sigma)\rceil$ bits of advice are sufficient for optimality.
$(n - 2\,\mathbf{Opt}(\sigma))\log\mathbf{Opt}(\sigma)$ bits of advice are required to guarantee optimality.

Breaking the Lower Bound — Effectively

Recall: All online algorithms have a competitive ratio of at least 1.54.

Breaking the Lower Bound — Effectively

Recall: All online algorithms have a competitive ratio of at least 1.54.

Advice of size $\lceil \log n \rceil$ is sufficient to achieve a competitive ratio of 1.5.

Breaking the Lower Bound — Effectively

Recall: All online algorithms have a competitive ratio of at least 1.54.

Advice of size $\lceil \log n \rceil$ is sufficient to achieve a competitive ratio of 1.5.

Advice: The number of items in range $(1/2, 2/3]$.

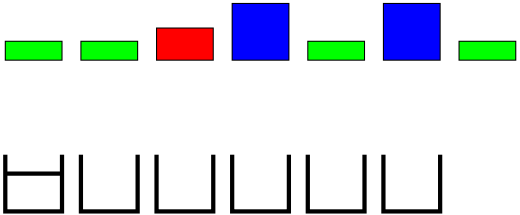
Breaking the Lower Bound — Effectively

Recall: All online algorithms have a competitive ratio of at least 1.54.

Advice of size $\lceil \log n \rceil$ is sufficient to achieve a competitive ratio of 1.5.

Advice: The number of items in range $(1/2, 2/3]$.
Algorithm: Reserve a space of size $2/3$ for each of them
Apply First-Fit for the other items.

Advice: 1



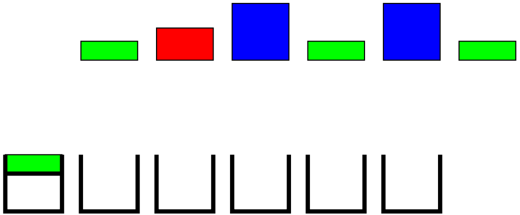
Breaking the Lower Bound — Effectively

Recall: All online algorithms have a competitive ratio of at least 1.54.

Advice of size $\lceil \log n \rceil$ is sufficient to achieve a competitive ratio of 1.5.

Advice: The number of items in range $(1/2, 2/3]$.
Algorithm: Reserve a space of size $2/3$ for each of them
Apply First-Fit for the other items.

Advice: 1

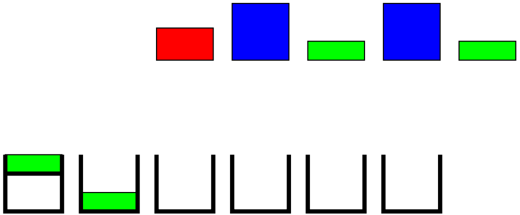


Recall: All online algorithms have a competitive ratio of at least 1.54.

Advice of size $\lceil \log n \rceil$ is sufficient to achieve a competitive ratio of 1.5.

Advice: The number of items in range $(1/2, 2/3]$.
Algorithm: Reserve a space of size $2/3$ for each of them
Apply First-Fit for the other items.

Advice: 1

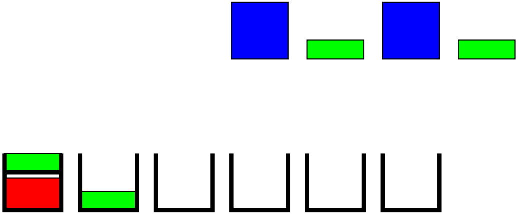


Recall: All online algorithms have a competitive ratio of at least 1.54.

Advice of size $\lceil \log n \rceil$ is sufficient to achieve a competitive ratio of 1.5.

Advice: The number of items in range $(1/2, 2/3]$.
Algorithm: Reserve a space of size $2/3$ for each of them
Apply First-Fit for the other items.

Advice: 1



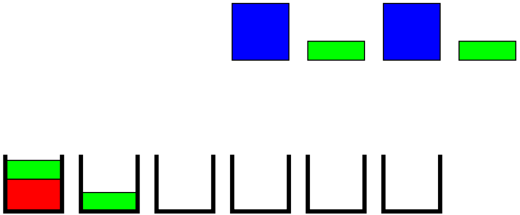
Breaking the Lower Bound — Effectively

Recall: All online algorithms have a competitive ratio of at least 1.54.

Advice of size $\lceil \log n \rceil$ is sufficient to achieve a competitive ratio of 1.5.

Advice: The number of items in range $(1/2, 2/3]$.
Algorithm: Reserve a space of size $2/3$ for each of them
Apply First-Fit for the other items.

Advice: 1



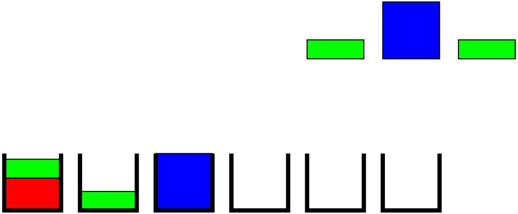
Breaking the Lower Bound — Effectively

Recall: All online algorithms have a competitive ratio of at least 1.54.

Advice of size $\lceil \log n \rceil$ is sufficient to achieve a competitive ratio of 1.5.

Advice: The number of items in range $(1/2, 2/3]$.
Algorithm: Reserve a space of size $2/3$ for each of them
Apply First-Fit for the other items.

Advice: 1



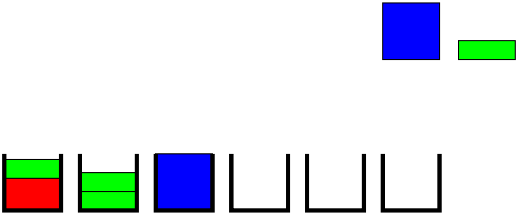
Breaking the Lower Bound — Effectively

Recall: All online algorithms have a competitive ratio of at least 1.54.

Advice of size $\lceil \log n \rceil$ is sufficient to achieve a competitive ratio of 1.5.

Advice: The number of items in range $(1/2, 2/3]$.
Algorithm: Reserve a space of size $2/3$ for each of them
Apply First-Fit for the other items.

Advice: 1



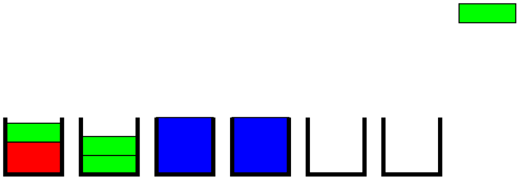
Breaking the Lower Bound — Effectively

Recall: All online algorithms have a competitive ratio of at least 1.54.

Advice of size $\lceil \log n \rceil$ is sufficient to achieve a competitive ratio of 1.5.

Advice: The number of items in range $(1/2, 2/3]$.
Algorithm: Reserve a space of size $2/3$ for each of them
Apply First-Fit for the other items.

Advice: 1

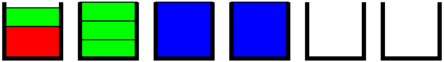


Recall: All online algorithms have a competitive ratio of at least 1.54.

Advice of size $\lceil \log n \rceil$ is sufficient to achieve a competitive ratio of 1.5.

Advice: The number of items in range $(1/2, 2/3]$.
Algorithm: Reserve a space of size $2/3$ for each of them
Apply First-Fit for the other items.

Advice: 1



Advice of Linear Size

An online algorithm which receives 2 bits of advice per request (plus an additive lower order term).

Advice of Linear Size

An online algorithm which receives 2 bits of advice per request (plus an additive lower order term).

Achieves a competitive ratio of $4/3 + \varepsilon$, for any positive value of ε.

Advice of Linear Size

An online algorithm which receives 2 bits of advice per request (plus an additive lower order term).

Achieves a competitive ratio of $4/3 + \varepsilon$, for any positive value of ε.

A variety of bin packing techniques are used in the proof.

Advice of Linear Size

An online algorithm which receives 2 bits of advice per request (plus an additive lower order term).

Achieves a competitive ratio of $4/3 + \varepsilon$, for any positive value of ε.

A variety of bin packing techniques are used in the proof.

Advice depends on $\textsc{Opt}$'s packing.

A Lower Bound

A linear amount of advice is required to achieve a competitive ratio better than $9/8$.

Get a trade-off — better ratio requires more advice

A Lower Bound

A linear amount of advice is required to achieve a competitive ratio better than $9/8$.

Get a trade-off — better ratio requires more advice

Reduction order:
Binary string guessing problem $\longrightarrow$ Binary separation problem
Binary separation problem $\longrightarrow$ Bin packing problem

Binary String Guessing Problem

Binary string guessing problem (with known history): 2-SGKH
[Emek,Fraigniaud,Korman,Rosén, 2011]
[Böckenhauer,Hromkovic,Komm,Krug,Smula,Sprock, 2013]

Binary String Guessing Problem

Binary string guessing problem (with known history): 2-SGKH
[Emek,Fraigniaud,Korman,Rosén, 2011]
[Böckenhauer,Hromkovic,Komm,Krug,Smula,Sprock, 2013]

- Guess the next bit in a bit string revealed in an online manner

Binary String Guessing Problem

Binary string guessing problem (with known history): 2-SGKH
[Emek,Fraigniaud,Korman,Rosén, 2011]
[Böckenhauer,Hromkovic,Komm,Krug,Smula,Sprock, 2013]

- Guess the next bit in a bit string revealed in an online manner
- $\langle 0,\ 1,\ 0,\ ?\rangle$

Binary String Guessing Problem

Binary string guessing problem (with known history): 2-SGKH
[Emek,Fraigniaud,Korman,Rosén, 2011]
[Böckenhauer,Hromkovic,Komm,Krug,Smula,Sprock, 2013]

- Guess the next bit in a bit string revealed in an online manner
- $\langle 0,\ 1,\ 0,\ ?\rangle$
- A linear amount advice is required to correctly guess more than half of the bits.

Binary String Guessing Problem

Binary string guessing problem (with known history): 2-SGKH
[Emek,Fraigniaud,Korman,Rosén, 2011]
[Böckenhauer,Hromkovic,Komm,Krug,Smula,Sprock, 2013]

- Guess the next bit in a bit string revealed in an online manner
- $\langle 0,\ 1,\ 0,\ ?\rangle$
- A linear amount advice is required to correctly guess more than half of the bits.

Theorem

On inputs of length n, any deterministic algorithm for 2-SGKH that is guaranteed to guess correctly on more than αn bits, for $1/2 \leq \alpha < 1$, needs to read at least $(1 + (1 - \alpha)\log(1 - \alpha) + \alpha\log(\alpha))n$ bits of advice.

Note: If we assume the number, n_0, of 0s is given, we need at least $(1 + (1 - \alpha)\log(1 - \alpha) + \alpha\log(\alpha))n - e(n_0)$ bits of advice, where $e(n_0) = \lceil\log(n_0 + 1)\rceil + 2\lceil\log(\lceil\log(n_0 + 1)\rceil + 1)\rceil + 1$ (self-delimiting code).

Binary Separation Problem

Binary separation problem:

- For a sequence of $n_1 + n_2$ items decide whether an item belongs to the n_1 smaller items or n_2 larger items.

Binary Separation Problem

Binary separation problem:

- For a sequence of $n_1 + n_2$ items decide whether an item belongs to the n_1 smaller items or n_2 larger items.
- $\left\langle \frac{1}{2}\ (s),\ \frac{3}{4}\ (l),\ \frac{5}{8}\ (s),\ \frac{11}{16}\ (?) \right\rangle$

Binary Separation Problem

Binary separation problem:

- For a sequence of $n_1 + n_2$ items decide whether an item belongs to the n_1 smaller items or n_2 larger items.
- $\left\langle \frac{1}{2}\ (s),\ \frac{3}{4}\ (l),\ \frac{5}{8}\ (s),\ \frac{11}{16}\ (?) \right\rangle$
- Don't have to choose in $[0, 1]$.

Binary Separation Problem

Binary separation problem:

- For a sequence of $n_1 + n_2$ items decide whether an item belongs to the n_1 smaller items or n_2 larger items.
- $\left\langle \frac{1}{2}\ (s),\ \frac{3}{4}\ (l),\ \frac{5}{8}\ (s),\ \frac{11}{16}\ (?) \right\rangle$
- Don't have to choose in $[0, 1]$.
- Don't have to choose the exact middle value.

Reduction from Binary separation to Bin packing

Idea: Create small and large items, so $\textsc{Alg}$ has to decide which is which.

Reduction from Binary separation to Bin packing

Idea: Create small and large items, so $\textsc{Alg}$ has to decide which is which.

Give n_2 items of size $\frac{1}{2} + \epsilon$ — begin items, B.

Reduction from Binary separation to Bin packing

Idea: Create small and large items, so ALG has to decide which is which.

Give n_2 items of size $\frac{1}{2} + \epsilon$ — begin items, B.
ALG (and OPT) must put them in separate bins.

Reduction from Binary separation to Bin packing

Idea: Create small and large items, so ALG has to decide which is which.

Give n_2 items of size $\frac{1}{2} + \epsilon$ — begin items, B.
ALG (and OPT) must put them in separate bins.
Give large items, L and small items, S:

- OPT places large items with begin items.
- OPT places small items, one per bin.
- ALG much choose.

Reduction from Binary separation to Bin packing

Idea: Create small and large items, so ALG has to decide which is which.

Give n_2 items of size $\frac{1}{2} + \epsilon$ — begin items, B.
ALG (and OPT) must put them in separate bins.
Give large items, L and small items, S:

- OPT places large items with begin items.
- OPT places small items, one per bin.
- ALG much choose.

For each small item of size $\frac{1}{2} - \epsilon_i$,
give an item of size $\frac{1}{2} + \epsilon_i$ — matching items, M.

Reduction from Binary separation to Bin packing

Idea: Create small and large items, so ALG has to decide which is which.

Give n_2 items of size $\frac{1}{2} + \epsilon$ — begin items, B.
ALG (and OPT) must put them in separate bins.
Give large items, L and small items, S:

- OPT places large items with begin items.
- OPT places small items, one per bin.
- ALG much choose.

For each small item of size $\frac{1}{2} - \epsilon_i$,
give an item of size $\frac{1}{2} + \epsilon_i$ — matching items, M.
OPT packs matching items with small items, using $n_1 + n_2$ bins.

Reduction from Binary separation to Bin packing

Large item $+$ matching item > 1.

Reduction from Binary separation to Bin packing

Large item + matching item > 1.

A large item may not be with a begin item because

- bad guess for that item
- bad guess for small item — no space

Reduction from Binary separation to Bin packing

Large item $+$ matching item > 1.

A large item may not be with a begin item because

- bad guess for that item
- bad guess for small item — no space

Let
$x = \max\{\text{number bad guesses for small}, \text{number bad guesses for large}\}$

Reduction from Binary separation to Bin packing

Large item $+$ matching item $>$ 1.

A large item may not be with a begin item because

- bad guess for that item
- bad guess for small item — no space

Let
$x = \max\{\text{number bad guesses for small}, \text{number bad guesses for large}\}$
x large items not paired with begin items.

Reduction from Binary separation to Bin packing

Large item + matching item $>$ 1.

A large item may not be with a begin item because

- bad guess for that item
- bad guess for small item — no space

Let
$x = \max\{\text{number bad guesses for small}, \text{number bad guesses for large}\}$
x large items not paired with begin items.
At most 2 fit in a bin together.

Reduction from Binary separation to Bin packing

Large item $+$ matching item > 1.

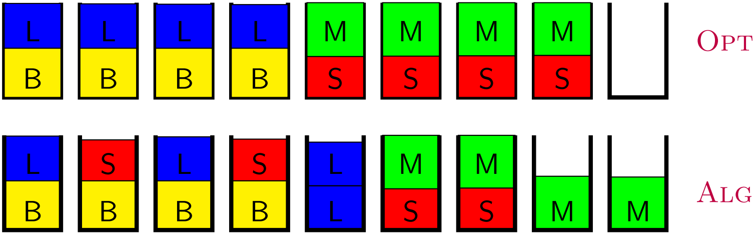
A large item may not be with a begin item because

- bad guess for that item
- bad guess for small item — no space

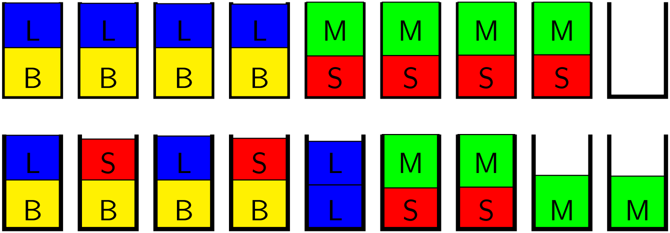
Let
$x = \max\{\text{number bad guesses for small}, \text{number bad guesses for large}\}$
x large items not paired with begin items.
At most 2 fit in a bin together.

4 errors in binary separation $\Rightarrow \; \geq 1$ more bin

Reduction from Binary separation to Bin packing



Reduction from Binary separation to Bin packing



OPT Result: 8

ALG Result: 9

Lower bound result for bin packing

Theorem

On inputs of length n, to achieve a competitive ratio of c $(1 < c < 9/8)$*, an online algorithm must get at least*
$(1 + (4c-4)\log(4c-4) + (5-4c)\log(5-4c))n - e(n)$ *bits of advice.*

Recall that $e(n) = \lceil \log(n+1) \rceil + 2\lceil \log(\lceil \log(n+1) \rceil + 1) \rceil + 1$.

[Renault, Rosén, van Stee, 2013] For a fixed competitive ratio, there exists an online algorithm which only needs linear advice:
They present an algorithm for online bin packing which is $(1+3\delta)$-competitive (or asymptotically $(1+2\delta)$-competitive), using $s = \frac{1}{\delta} \log \frac{2}{\delta^2} + \log \frac{2}{\delta^2} + 3$ bits of advice per request.

Section 3

Open problems

Open Problems

- Linear advice is needed to be c-competitive, $c < 9/8$.
  Linear advice is sufficient for any fixed c. There is a huge gap, though.
- $(2 + o(1))n$ advice is sufficient to be $4/3 + \epsilon$-competitive.
  Can one get a better ratio with so few bits?
- $O(\log n)$ advice is sufficient to be $3/2$-competitive.
  How many bits are required to break the 1.54 lower bound?

Thank you for your attention.

Reduction from 2-SGKH to Binary separation

```
small = 0; large = 1
repeat
    mid = (large − small) / 2
    class_guess = SeparationAlgorithm.ClassifyThis(mid)
    if class_guess = “large” then
        bit_guess = 0
    else
        bit_guess = 1
    actual_bit = Guess(bit_guess) {The actual value is received after guessing (2-SGKH).}
    if actual_bit = 0 then
        large = mid {We let “large” be the correct decision.}
    else
        small = mid {We let “small” be the correct decision.}
until end of sequence
```