Online Bin Packing with Predictions

Bin Packing Seminar Series

Shahin Kamali

April 14, 2020

University of Manitoba

Joint work S. Angelopoulos, J. Boyar, C. Dürr, S. Jin, K. S. Larsen,

A. López-Ortiz, M. P. Renault, A. Rosén, K. Shadkami

Overview

- Question: How online algorithms can benefit from some advice/prediction about the input sequence?

Overview

- Question: How online algorithms can benefit from some advice/prediction about the input sequence?
- We consider general frameworks of **advice** and **predictions**.

Part I: Modeling Prediction

https://www.cnbc.com/2020/04/10/coronavirus-empty-streets-around-the-world-are-attracting-wildlife.html

Relaxing the Online Constraint

- Relax the online constraint via some information about input.
    - Lookahead? closed online problems? locality?

Relaxing the Online Constraint

- Relax the online constraint via some information about input.
  - Lookahead? closed online problems? locality?
- **Advice model**: give **any** information about the input sequence.
  - The main constraint is the **advice size**.

Relaxing the Online Constraint

- Relax the online constraint via some information about input.
  - Lookahead? closed online problems? locality?
- **Advice model**: give **any** information about the input sequence.
  - The main constraint is the **advice size**.
- The advice scheme indicates:
  - What the advice should be.
  - How an algorithm should work, given specific advice.

Advice Example: Ski Rental

- Ski-rental problem: we go skiing for an unknown number U of days:
  - At each day rent the equipment at a cost of 1 or buy it once at a cost of B ($B > 1$).

Advice Example: Ski Rental

- Ski-rental problem: we go skiing for an unknown number U of days:
  - At each day rent the equipment at a cost of 1 or buy it once at a cost of B ($B > 1$).
- One bit of advice: is $B < U$?
  - If yes, buy on day 1; otherwise, always rent.

Advice Example: Ski Rental

- Ski-rental problem: we go skiing for an unknown number U of days:
  - At each day rent the equipment at a cost of 1 or buy it once at a cost of B ($B > 1$).
- One bit of advice: is $B < U$?
  - If yes, buy on day 1; otherwise, always rent.
  - With 1 bit of advice, one can achieve an optimal solution.

Advice Example: Bin Packing

- The advice size indicates the competitive ratio.
  - The larger is the advice, the better is the competitive ratio.

Advice Example: Bin Packing

- The advice size indicates the competitive ratio.
  - The larger is the advice, the better is the competitive ratio.
- Advice for bin packing:
  - Encode the optimal packing in $O(n \log N)$ bits (N is the cost of Opt).

Advice Example: Bin Packing

- The advice size indicates the competitive ratio.
  - The larger is the advice, the better is the competitive ratio.
- Advice for bin packing:
  - Encode the optimal packing in $O(n \log N)$ bits (N is the cost of Opt).
  - One cannot achieve an optimal solution with an asymptotically smaller number of bits [Boyar et al., 2016].

Advice Example: Bin Packing

- The advice size indicates the competitive ratio.
  - The larger is the advice, the better is the competitive ratio.
- Advice for bin packing:
  - Encode the optimal packing in $O(n \log N)$ bits (N is the cost of Opt).
  - One cannot achieve an optimal solution with an asymptotically smaller number of bits [Boyar et al., 2016].
- What can be done with a smaller advice, e.g., of size $O(\log n)$?

Breaking the Lower Bound

- Consider **ReserveCritical** algorithm[Boyar et al., 2016].
- Receive the number of **critical** items, in the range $(1/2, 2/3])$, with $O(\log n)$ bits of advice.

Breaking the Lower Bound

- Consider **ReserveCritical** algorithm[Boyar et al., 2016].
- Receive the number of **critical** items, in the range $(1/2, 2/3])$, with $O(\log n)$ bits of advice.
  - At the beginning, reserve a space of size $2/3$ for critical items

Breaking the Lower Bound

- Consider **ReserveCritical** algorithm[Boyar et al., 2016].
- Receive the number of **critical** items, in the range $(1/2, 2/3])$, with $O(\log n)$ bits of advice.
  - At the beginning, reserve a space of size $2/3$ for critical items
  - **huge** items (of size $> 2/3$): open a new bin
  - **critical** items (of size $\in (1/2, 2/3]$): place in a reserve space
  - **mini** item (of size in $(1/3, 1/2]$): place two of them in the same bin
  - **tiny** items (of size $< 1/3$): apply First-Fit to place in bins with critical or other tiny items

ReserveCritical Algorithm

- At the beginning, reserve a space of size 2/3 for critical items
  - **huge** items: open a new bin (no other item goes there)
  - **critical** items: place in a reserve space
  - **mini** item: place two of them in the same bin
  - **tiny** items: apply FirstFit to place in critical or tiny bins

$\sigma = \langle$ 0.3 0.9 0.6 0.5 0.1 0.1 0.56 0.4 0.3 0.45 0.8 0.51 0.41 0.2 0.1 0.37 0.3 $\rangle$

ReserveCritical Algorithm

- At the beginning, reserve a space of size 2/3 for critical items
  - **huge** items: open a new bin (no other item goes there)
  - **critical** items: place in a reserve space
  - **mini** item: place two of them in the same bin
  - **tiny** items: apply FirstFit to place in critical or tiny bins

$\sigma = \langle$ 0.3 0.9 0.6 0.5 0.1 0.1 0.56 0.4 0.3 0.45 0.8 0.51 0.41 0.2 0.1 0.37 0.3 $\rangle$

ReserveCritical Algorithm

- At the beginning, reserve a space of size 2/3 for critical items
  - **huge** items: open a new bin (no other item goes there)
  - **critical** items: place in a reserve space
  - **mini** item: place two of them in the same bin
  - **tiny** items: apply FirstFit to place in critical or tiny bins

$\sigma = \langle$ 0.3 0.9 0.6 0.5 0.1 0.1 0.56 0.4 0.3 0.45 0.8 0.51 0.41 0.2 0.1 0.37 0.3 $\rangle$

ReserveCritical Algorithm

- At the beginning, reserve a space of size 2/3 for critical items
  - **huge** items: open a new bin (no other item goes there)
  - **critical** items: place in a reserve space
  - **mini** item: place two of them in the same bin
  - **tiny** items: apply FirstFit to place in critical or tiny bins

$\sigma = \langle$ 0.3 0.9 0.6 0.5 0.1 0.1 0.56 0.4 0.3 0.45 0.8 0.51 0.41 0.2 0.1 0.37 0.3 $\rangle$

ReserveCritical Algorithm

- At the beginning, reserve a space of size 2/3 for critical items
  - **huge** items: open a new bin (no other item goes there)
  - **critical** items: place in a reserve space
  - **mini** item: place two of them in the same bin
  - **tiny** items: apply FirstFit to place in critical or tiny bins

$\sigma = \langle$ 0.3 0.9 0.6 0.5 0.1 0.1 0.56 0.4 0.3 0.45 0.8 0.51 0.41 0.2 0.1 0.37 0.3 $\rangle$

ReserveCritical Algorithm

- At the beginning, reserve a space of size 2/3 for critical items
  - **huge** items: open a new bin (no other item goes there)
  - **critical** items: place in a reserve space
  - **mini** item: place two of them in the same bin
  - **tiny** items: apply FirstFit to place in critical or tiny bins

$\sigma = \langle$ 0.3 0.9 0.6 0.5 0.1 0.1 0.56 0.4 0.3 0.45 0.8 0.51 0.41 0.2 0.1 0.37 0.3 $\rangle$

ReserveCritical Algorithm

- At the beginning, reserve a space of size 2/3 for critical items
  - **huge** items: open a new bin (no other item goes there)
  - **critical** items: place in a reserve space
  - **mini** item: place two of them in the same bin
  - **tiny** items: apply FirstFit to place in critical or tiny bins

$\sigma = \langle$ 0.3 0.9 0.6 0.5 0.1 0.1 0.56 0.4 0.3 0.45 0.8 0.51 0.41 0.2 0.1 0.37 0.3 $\rangle$

ReserveCritical Algorithm

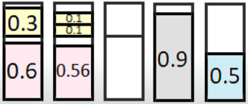
- At the beginning, reserve a space of size 2/3 for critical items
  - **huge** items: open a new bin (no other item goes there)
  - **critical** items: place in a reserve space
  - **mini** item: place two of them in the same bin
  - **tiny** items: apply FirstFit to place in critical or tiny bins

$\sigma = \langle$ 0.3 0.9 0.6 0.5 0.1 0.1 0.56 0.4 0.3 0.45 0.8 0.51 0.41 0.2 0.1 0.37 0.3 $\rangle$

ReserveCritical Algorithm

- At the beginning, reserve a space of size 2/3 for critical items
  - **huge** items: open a new bin (no other item goes there)
  - **critical** items: place in a reserve space
  - **mini** item: place two of them in the same bin
  - **tiny** items: apply FirstFit to place in critical or tiny bins

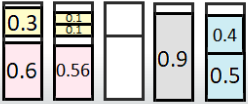
$\sigma = \langle$ 0.3 0.9 0.6 0.5 0.1 0.1 0.56 0.4 0.3 0.45 0.8 0.51 0.41 0.2 0.1 0.37 0.3 $\rangle$



ReserveCritical Algorithm

- At the beginning, reserve a space of size 2/3 for critical items
  - **huge** items: open a new bin (no other item goes there)
  - **critical** items: place in a reserve space
  - **mini** item: place two of them in the same bin
  - **tiny** items: apply FirstFit to place in critical or tiny bins

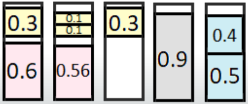
$\sigma = \langle$ 0.3 0.9 0.6 0.5 0.1 0.1 0.56 0.4 0.3 0.45 0.8 0.51 0.41 0.2 0.1 0.37 0.3 $\rangle$



ReserveCritical Algorithm

- At the beginning, reserve a space of size 2/3 for critical items
    - **huge** items: open a new bin (no other item goes there)
    - **critical** items: place in a reserve space
    - **mini** item: place two of them in the same bin
    - **tiny** items: apply FirstFit to place in critical or tiny bins

$\sigma = \langle$ 0.3 0.9 0.6 0.5 0.1 0.1 0.56 0.4 0.3 0.45 0.8 0.51 0.41 0.2 0.1 0.37 0.3 $\rangle$



ReserveCritical Algorithm

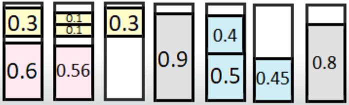
- At the beginning, reserve a space of size 2/3 for critical items
  - **huge** items: open a new bin (no other item goes there)
  - **critical** items: place in a reserve space
  - **mini** item: place two of them in the same bin
  - **tiny** items: apply FirstFit to place in critical or tiny bins

$\sigma = \langle$ 0.3 0.9 0.6 0.5 0.1 0.1 0.56 0.4 0.3 0.45 0.8 0.51 0.41 0.2 0.1 0.37 0.3 $\rangle$

ReserveCritical Algorithm

- At the beginning, reserve a space of size 2/3 for critical items
  - **huge** items: open a new bin (no other item goes there)
  - **critical** items: place in a reserve space
  - **mini** item: place two of them in the same bin
  - **tiny** items: apply FirstFit to place in critical or tiny bins

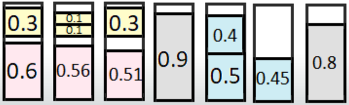
$\sigma = \langle$ 0.3 0.9 0.6 0.5 0.1 0.1 0.56 0.4 0.3 0.45 0.8 0.51 0.41 0.2 0.1 0.37 0.3 $\rangle$



ReserveCritical Algorithm

- At the beginning, reserve a space of size 2/3 for critical items
  - **huge** items: open a new bin (no other item goes there)
  - **critical** items: place in a reserve space
  - **mini** item: place two of them in the same bin
  - **tiny** items: apply FirstFit to place in critical or tiny bins

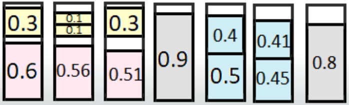
$\sigma = \langle$ 0.3 0.9 0.6 0.5 0.1 0.1 0.56 0.4 0.3 0.45 0.8 0.51 0.41 0.2 0.1 0.37 0.3 $\rangle$



ReserveCritical Algorithm

- At the beginning, reserve a space of size 2/3 for critical items
  - **huge** items: open a new bin (no other item goes there)
  - **critical** items: place in a reserve space
  - **mini** item: place two of them in the same bin
  - **tiny** items: apply FirstFit to place in critical or tiny bins

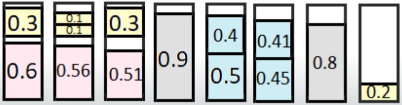
$\sigma = \langle$ 0.3 0.9 0.6 0.5 0.1 0.1 0.56 0.4 0.3 0.45 0.8 0.51 0.41 0.2 0.1 0.37 0.3 $\rangle$



ReserveCritical Algorithm

- At the beginning, reserve a space of size 2/3 for critical items
  - **huge** items: open a new bin (no other item goes there)
  - **critical** items: place in a reserve space
  - **mini** item: place two of them in the same bin
  - **tiny** items: apply FirstFit to place in critical or tiny bins

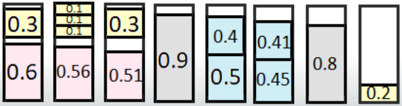
$\sigma = \langle$ 0.3 0.9 0.6 0.5 0.1 0.1 0.56 0.4 0.3 0.45 0.8 0.51 0.41 0.2 0.1 0.37 0.3 $\rangle$



ReserveCritical Algorithm

- At the beginning, reserve a space of size 2/3 for critical items
  - **huge** items: open a new bin (no other item goes there)
  - **critical** items: place in a reserve space
  - **mini** item: place two of them in the same bin
  - **tiny** items: apply FirstFit to place in critical or tiny bins

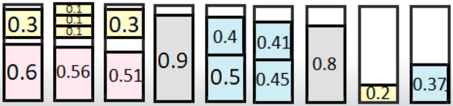
$\sigma = \langle$ 0.3 0.9 0.6 0.5 0.1 0.1 0.56 0.4 0.3 0.45 0.8 0.51 0.41 0.2 0.1 0.37 0.3 $\rangle$



ReserveCritical Algorithm

- At the beginning, reserve a space of size 2/3 for critical items
  - **huge** items: open a new bin (no other item goes there)
  - **critical** items: place in a reserve space
  - **mini** item: place two of them in the same bin
  - **tiny** items: apply FirstFit to place in critical or tiny bins

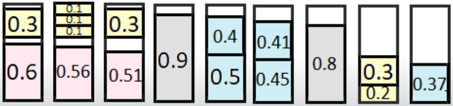
$\sigma = \langle$ 0.3 0.9 0.6 0.5 0.1 0.1 0.56 0.4 0.3 0.45 0.8 0.51 0.41 0.2 0.1 0.37 0.3 $\rangle$



ReserveCritical Algorithm

- At the beginning, reserve a space of size 2/3 for critical items
    - **huge** items: open a new bin (no other item goes there)
    - **critical** items: place in a reserve space
    - **mini** item: place two of them in the same bin
    - **tiny** items: apply FirstFit to place in critical or tiny bins

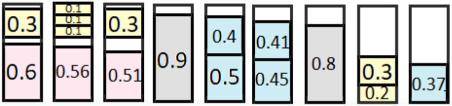
$\sigma = \langle$ 0.3 0.9 0.6 0.5 0.1 0.1 0.56 0.4 0.3 0.45 0.8 0.51 0.41 0.2 0.1 0.37 0.3 $\rangle$



ReserveCritical Algorithm

- At the beginning, reserve a space of size 2/3 for critical items
  - **huge** items: open a new bin (no other item goes there)
  - **critical** items: place in a reserve space
  - **mini** item: place two of them in the same bin
  - **tiny** items: apply FirstFit to place in critical or tiny bins

$\sigma = \langle$ 0.3 0.9 0.6 0.5 0.1 0.1 0.56 0.4 0.3 0.45 0.8 0.51 0.41 0.2 0.1 0.37 0.3 $\rangle$



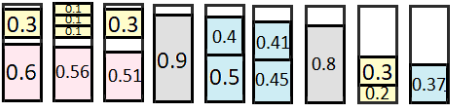
Theorem

The competitive ratio of ReserveCritical is at most 1.5.

ReserveCritical Algorithm

- At the beginning, reserve a space of size 2/3 for critical items
  - **huge** items: open a new bin (no other item goes there)
  - **critical** items: place in a reserve space
  - **mini** item: place two of them in the same bin
  - **tiny** items: apply FirstFit to place in critical or tiny bins

$\sigma = \langle$ 0.3 0.9 0.6 0.5 0.1 0.1 0.56 0.4 0.3 0.45 0.8 0.51 0.41 0.2 0.1 0.37 0.3 $\rangle$



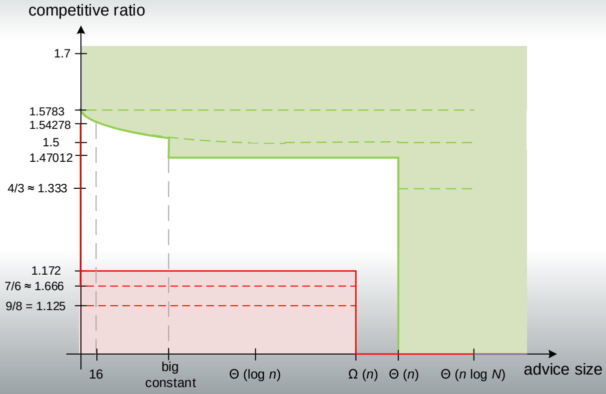
Theorem

The competitive ratio of ReserveCritical is at most 1.5.

- Instead of the number of critical items in $O(\log n)$, one can encode $\gamma = \frac{no.\ critical\ bin}{no.\ critical\ bins + no.\ small\ bins}$ in $O(1)$ [Angelopoulos et al., 2018].

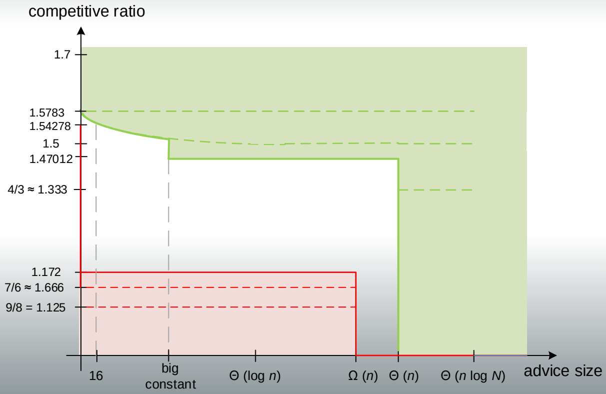
Bin Packing & Advice

- No advice: best upper and lower bounds by [Balogh et al., 2018] and [Balogh et al., 2019].



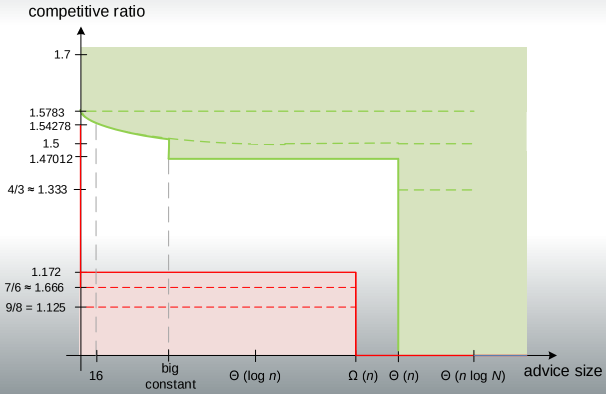
Bin Packing & Advice

- With $k \geq 4$ bits, one can get a competitive ratio of $1.5 + \frac{15}{2^{k/2}+1}$ [Angelopoulos et al., 2018].



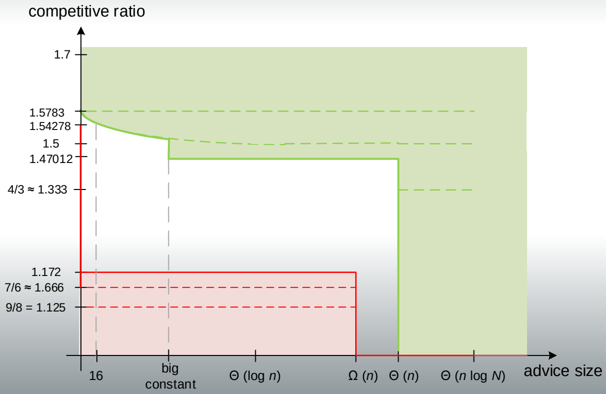
Bin Packing & Advice

- With $O(1)$ bits, one can get a competitive ratio of 1.4702 [Angelopoulos et al., 2018].



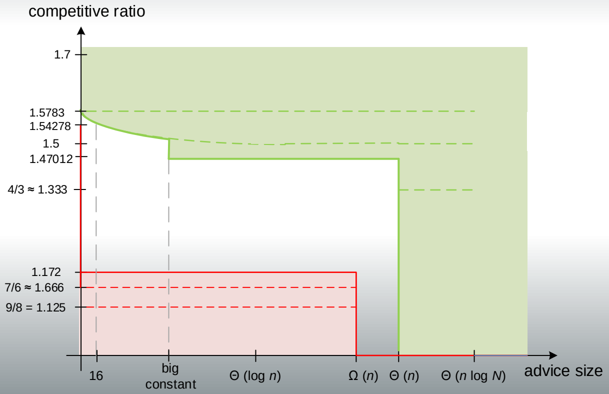
Bin Packing & Advice

- With linear number bits, one can achieve a competitive ratio of 4/3 [Boyar et al., 2016].



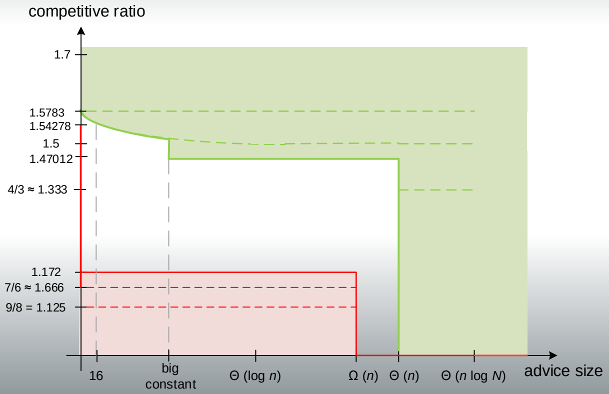
Bin Packing & Advice

- With a linear number bits, one can achieve a competitive ratio of 1.0 [Renault et al., 2015].



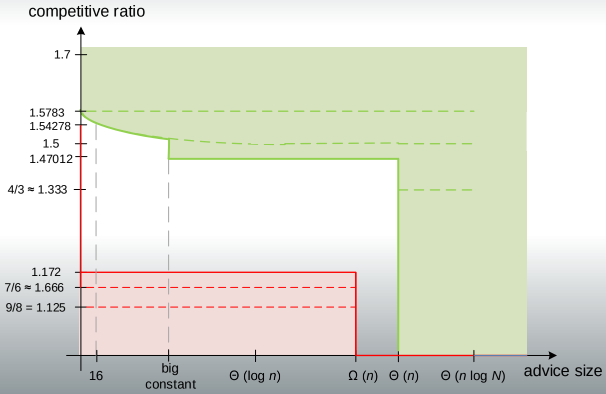
Bin Packing & Advice

- For a competitive ratio better than 9/8, a linear number of bits are required [Boyar et al., 2016].



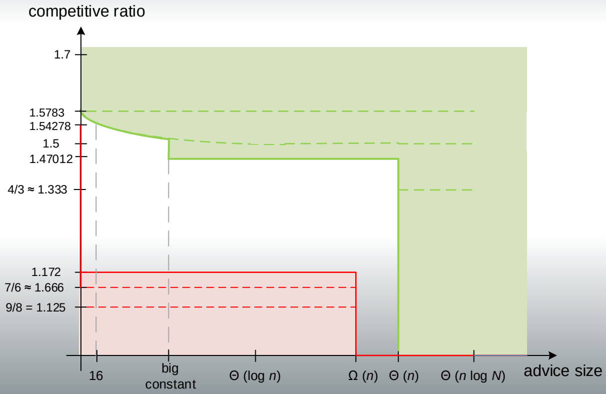
Bin Packing & Advice

- For a competitive ratio better than 7/6, a linear number of bits are required [Angelopoulos et al., 2018].



Bin Packing & Advice

- For a competitive ratio better than $4 - 2\sqrt{2} \approx 1.172$, a linear number of bits are required [Mikkelsen, 2016].



Advice Model in Practice

- In practice:
  - The advice size is not the main concern.

Advice Model in Practice

- In practice:
  - The advice size is not the main concern.
  - The advice cannot be designed to be anything; it should be predictable.

Advice Model in Practice

- In practice:
  - The advice size is not the main concern.
  - The advice cannot be designed to be anything; it should be predictable.
  - Predictions are often noisy.

Online Algorithms with Prediction

- Predictions about the input are given (e.g., item frequencies in bin packing).
- There is an error η in prediction.
  - It is desirable to state the competitive ratio as a function of error.
  - When $\eta = 0$, the competitive ratio is called **consistency**.
  - When η takes its largest value, the competitive ratio is called **robustness**.

Ski-rental with Prediction

- Prediction: the number U' of skiing days.

Ski-rental with Prediction

- Prediction: the number U' of skiing days.
  - You cannot fully trust the prediction: if $B < U'$ and you buy at day 1, the robustness will be B (when $U = 1$).

Ski-rental with Prediction

- Prediction: the number U' of skiing days.
  - You cannot fully trust the prediction: if $B < U'$ and you buy at day 1, the robustness will be B (when $U = 1$).
- Algorithm A_λ ($\lambda \in [1, B]$) [Angelopoulos et al., 2020]:
  - If $B < U'$, then rent until day $\lambda - 1$ and buy on day λ; otherwise, buy on day B.
  - Small values of λ favor consistency and larger values favor robustness.

Ski-rental with Prediction

- Prediction: the number U' of skiing days.
  - You cannot fully trust the prediction: if $B < U'$ and you buy at day 1, the robustness will be B (when $U = 1$).
- Algorithm A_λ ($\lambda \in [1, B]$) [Angelopoulos et al., 2020]:
  - If $B < U'$, then rent until day $\lambda - 1$ and buy on day λ; otherwise, buy on day B.
  - Small values of λ favor consistency and larger values favor robustness.

Theorem

Algorithm A_λ has consistency $(1 + (\lambda - 1)/B)$ and robustness $1 + (B - 1)/\lambda$.

Ski-rental with Prediction

- Prediction: the number U' of skiing days.
  - You cannot fully trust the prediction: if $B < U'$ and you buy at day 1, the robustness will be B (when $U = 1$).
- Algorithm A_λ ($\lambda \in [1, B]$) [Angelopoulos et al., 2020]:
  - If $B < U'$, then rent until day $\lambda - 1$ and buy on day λ; otherwise, buy on day B.
  - Small values of λ favor consistency and larger values favor robustness.

Theorem

Algorithm A_λ has consistency $(1 + (\lambda - 1)/B)$ and robustness $1 + (B - 1)/\lambda$.

- Pareto-optimality: any algorithm with consistency $(1 + (\lambda - 1)/B)$ has robustness at least $1 + (B - 1)/\lambda$.

Online Algorithms with Predictions

- A different algorithm with competitive ratio $\min\{(1+\lambda)/\lambda, (1+\lambda)+\eta/((1-\lambda)Opt)\}$ [Purohit et al., 2018].
    - η is the error parameter, defined as $U - U'$, and $\lambda \in (0,1)$.

Online Algorithms with Predictions

- A different algorithm with competitive ratio $\min\{(1+\lambda)/\lambda, (1+\lambda)+\eta/((1-\lambda)Opt)\}$ [Purohit et al., 2018].
  - η is the error parameter, defined as $U - U'$, and $\lambda \in (0,1)$.
- Other online problems studied under the prediction model:
  - Online Bidding and List Update problem [Angelopoulos et al., 2020]
  - Contract Scheduling [Angelopoulos and Kamali, 2021].

Online Algorithms with Predictions

- A different algorithm with competitive ratio $\min\{(1+\lambda)/\lambda, (1+\lambda) + \eta/((1-\lambda)Opt)\}$ [Purohit et al., 2018].
    - η is the error parameter, defined as $U - U'$, and $\lambda \in (0, 1)$.
- Other online problems studied under the prediction model:
    - Online Bidding and List Update problem [Angelopoulos et al., 2020]
    - Contract Scheduling [Angelopoulos and Kamali, 2021].
    - Paging [Lykouris and Vassilvitskii, 2018, Rohatgi, 2020], Metric Task Systems [Antoniadis et al., 2020], scheduling [Lattanzi et al., 2020].

Part II: Bin Packing with Prediction

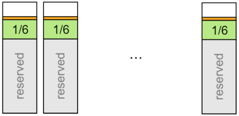
https://www.houseandgarden.co.uk/gallery/animals-cities-coronavirus-lockdown

ReserveCritical Revisit

- Prediction: $\gamma = \frac{\textit{no. critical bins}}{\textit{no. critical bins} + \textit{no. small bins}}$.
  - Maintain a ratio γ when opening bins with small items.

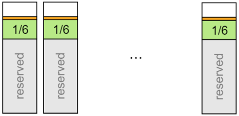
ReserveCritical Revisit

- Prediction: $\gamma = \frac{no.\ critical\ bins}{no.\ critical\ bins + no.\ small\ bins}$.
  - Maintain a ratio γ when opening bins with small items.
  - This scheme is not robust: assume $\gamma = 1$ and $\sigma = (1/6, \epsilon, 1/6, \epsilon, \ldots, 1/6, \epsilon)$; the c.r. is at least 6.



ReserveCritical Revisit

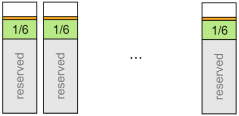
- Prediction: $\gamma = \frac{\textit{no. critical bins}}{\textit{no. critical bins} + \textit{no. small bins}}$.
  - Maintain a ratio γ when opening bins with small items.
  - This scheme is not robust: assume $\gamma = 1$ and $\sigma = (1/6, \epsilon, 1/6, \epsilon, \ldots, 1/6, \epsilon)$; the c.r. is at least 6.



- Instead of maintaining a ratio γ, maintain a ratio $\beta = \min\{\lambda, \gamma\}$, for some parameter λ.
  - E.g., when $\lambda = 0.5$, half of bins will be critical in the above example.

ReserveCritical Revisit

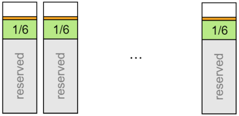
- Prediction: $\gamma = \frac{no.\ critical\ bins}{no.\ critical\ bins + no.\ small\ bins}$.
  - Maintain a ratio γ when opening bins with small items.
  - This scheme is not robust: assume $\gamma = 1$ and $\sigma = (1/6, \epsilon, 1/6, \epsilon, \ldots, 1/6, \epsilon)$; the c.r. is at least 6.



- Instead of maintaining a ratio γ, maintain a ratio $\beta = \min\{\lambda, \gamma\}$, for some parameter λ.
  - E.g., when $\lambda = 0.5$, half of bins will be critical in the above example.
  - The modified algorithm has consistency $1.5 + \frac{1-\lambda}{4-3\lambda}$, and robustness $1.5 + \max\{1/4, \frac{9\lambda}{8-6\lambda}\}$ [Angelopoulos et al., 2020].

ReserveCritical Revisit

- Prediction: $\gamma = \frac{\textit{no. critical bins}}{\textit{no. critical bins} + \textit{no. small bins}}$.
  - Maintain a ratio γ when opening bins with small items.
  - This scheme is not robust: assume $\gamma = 1$ and $\sigma = (1/6, \epsilon, 1/6, \epsilon, \ldots, 1/6, \epsilon)$; the c.r. is at least 6.



- Instead of maintaining a ratio γ, maintain a ratio $\beta = \min\{\lambda, \gamma\}$, for some parameter λ.
  - E.g., when $\lambda = 0.5$, half of bins will be critical in the above example.
  - The modified algorithm has consistency $1.5 + \frac{1-\lambda}{4-3\lambda}$, and robustness $1.5 + \max\{1/4, \frac{9\lambda}{8-6\lambda}\}$ [Angelopoulos et al., 2020].
  - One can get a r-consistent algorithm with robustness $\max\{33 - 18r, 7/4\}$ for any $r > 1.5$.

Bin Packing with Predictions in Practice

- Setting: items are integers in the range $[1..k]$, and bin capacity is k.

Bin Packing with Predictions in Practice

- Setting: items are integers in the range $[1..k]$, and bin capacity is k.
  - In the continuous setting, we cannot hope for consistency better than 1.172 unless predictions are of size $\Omega(n)$ [Mikkelsen, 2016].

Bin Packing with Predictions in Practice

- Setting: items are integers in the range $[1..k]$, and bin capacity is k.
  - In the continuous setting, we cannot hope for consistency better than 1.172 unless predictions are of size $\Omega(n)$ [Mikkelsen, 2016].
- Predictions: frequency of items of size x for any $x \in [1..k]$.
  - We use $f(x)$ as the actual frequencies, and $f'(x)$ as the predicted (noisy) predictions.

Profile-Packing Algorithm

- Profile-Packing with parameter M ($M \in O(1)$ is a large constant):
  - Form a **profile multiset** P in which there are $\lceil Mf'(x) \rceil$ items of size x.

Profile-Packing Algorithm

- Profile-Packing with parameter M ($M \in O(1)$ is a large constant):
  - Form a **profile multiset** P in which there are $\lceil Mf'(x) \rceil$ items of size x.
  - Form an optimal packing of P (a profile packing) in which there is a **placeholder** of size x for each item of size x in the profile multiset.

Profile-Packing Algorithm

- Profile-Packing with parameter M ($M \in O(1)$ is a large constant):
  - Form a **profile multiset** P in which there are $\lceil Mf'(x) \rceil$ items of size x.
  - Form an optimal packing of P (a profile packing) in which there is a **placeholder** of size x for each item of size x in the profile multiset.
  - Place each item in a placeholder of the same size (open a new profile packing if needed).

Profile-Packing Illustration

- Suppose predictions are perfect, i.e., $f(x) = f'(x)$ for all $x \in [1..k]$.
  - Assume $k = 10,\ M = 20$

x	1	2	3	4	5	6	7	8	9	10
$f(x) = f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.03	0

Profile-Packing Illustration

- Suppose predictions are perfect, i.e., $f(x) = f'(x)$ for all $x \in [1..k]$.
  - Assume $k = 10$, $M = 20$

x	1	2	3	4	5	6	7	8	9	10
$f(x) = f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.03	0

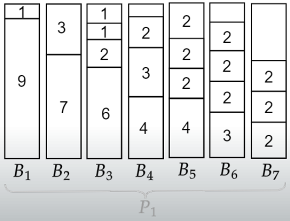
  - Profile multiset is $P = \{1^3, 2^{11}, 3^3, 4^2, 6, 7, 9\}$.
    - E.g., there are $\lceil 0.11 \cdot 20 \rceil = 3$ items of size 1 in the profile set.

Profile-Packing Illustration

- Suppose predictions are perfect, i.e., $f(x) = f'(x)$ for all $x \in [1..k]$.
  - Assume $k = 10$, $M = 20$

x	1	2	3	4	5	6	7	8	9	10
$f(x) = f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.03	0

  - Profile multiset is $P = \{1^3, 2^{11}, 3^3, 4^2, 6, 7, 9\}$.
    - E.g., there are $\lceil 0.11 \cdot 20 \rceil = 3$ items of size 1 in the profile set.

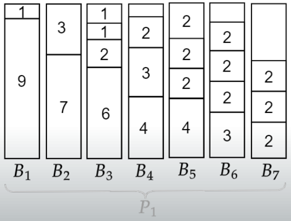


Profile-Packing Illustration

- Suppose predictions are perfect, i.e., $f(x) = f'(x)$ for all $x \in [1..k]$.
  - Assume $k = 10$, $M = 20$

x	1	2	3	4	5	6	7	8	9	10
$f(x) = f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.03	0

  - Profile multiset is $P = \{1^3, 2^{11}, 3^3, 4^2, 6, 7, 9\}$.
    - E.g., there are $\lceil 0.11 \cdot 20 \rceil = 3$ items of size 1 in the profile set.



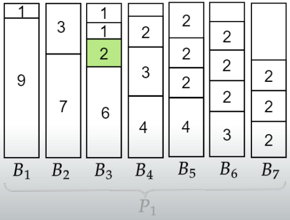
$\sigma = 2$

Profile-Packing Illustration

- Suppose predictions are perfect, i.e., $f(x) = f'(x)$ for all $x \in [1..k]$.
  - Assume $k = 10$, $M = 20$

x	1	2	3	4	5	6	7	8	9	10
$f(x) = f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.03	0

  - Profile multiset is $P = \{1^3, 2^{11}, 3^3, 4^2, 6, 7, 9\}$.
    - E.g., there are $\lceil 0.11 \cdot 20 \rceil = 3$ items of size 1 in the profile set.



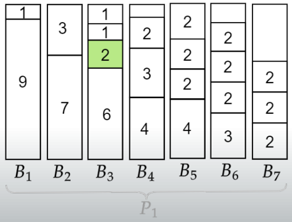
$\sigma = 2$

Profile-Packing Illustration

- Suppose predictions are perfect, i.e., $f(x) = f'(x)$ for all $x \in [1..k]$.
  - Assume $k = 10$, $M = 20$

x	1	2	3	4	5	6	7	8	9	10
$f(x) = f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.03	0

  - Profile multiset is $P = \{1^3, 2^{11}, 3^3, 4^2, 6, 7, 9\}$.
    - E.g., there are $\lceil 0.11 \cdot 20 \rceil = 3$ items of size 1 in the profile set.



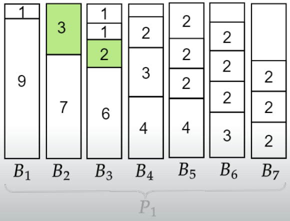
$\sigma = 2, 3$

Profile-Packing Illustration

- Suppose predictions are perfect, i.e., $f(x) = f'(x)$ for all $x \in [1..k]$.
  - Assume $k = 10$, $M = 20$

x	1	2	3	4	5	6	7	8	9	10
$f(x) = f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.03	0

  - Profile multiset is $P = \{1^3, 2^{11}, 3^3, 4^2, 6, 7, 9\}$.
    - E.g., there are $\lceil 0.11 \cdot 20 \rceil = 3$ items of size 1 in the profile set.



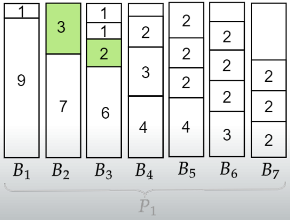
$\sigma = 2, 3$

Profile-Packing Illustration

- Suppose predictions are perfect, i.e., $f(x) = f'(x)$ for all $x \in [1..k]$.
  - Assume $k = 10$, $M = 20$

x	1	2	3	4	5	6	7	8	9	10
$f(x) = f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.03	0

  - Profile multiset is $P = \{1^3, 2^{11}, 3^3, 4^2, 6, 7, 9\}$.
    - E.g., there are $\lceil 0.11 \cdot 20 \rceil = 3$ items of size 1 in the profile set.



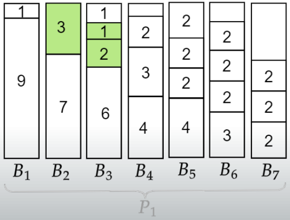
$\sigma = 2, 3, 1$

Profile-Packing Illustration

- Suppose predictions are perfect, i.e., $f(x) = f'(x)$ for all $x \in [1..k]$.
  - Assume $k = 10$, $M = 20$

x	1	2	3	4	5	6	7	8	9	10
$f(x) = f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.03	0

  - Profile multiset is $P = \{1^3, 2^{11}, 3^3, 4^2, 6, 7, 9\}$.
    - E.g., there are $\lceil 0.11 \cdot 20 \rceil = 3$ items of size 1 in the profile set.



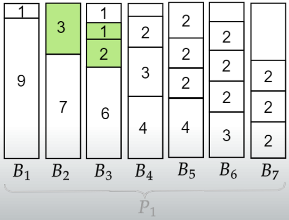
$\sigma = 2, 3, 1$

Profile-Packing Illustration

- Suppose predictions are perfect, i.e., $f(x) = f'(x)$ for all $x \in [1..k]$.
  - Assume $k = 10$, $M = 20$

x	1	2	3	4	5	6	7	8	9	10
$f(x) = f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.03	0

  - Profile multiset is $P = \{1^3, 2^{11}, 3^3, 4^2, 6, 7, 9\}$.
    - E.g., there are $\lceil 0.11 \cdot 20 \rceil = 3$ items of size 1 in the profile set.



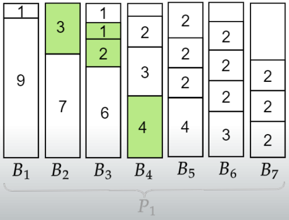
$\sigma = 2, 3, 1, 4$

Profile-Packing Illustration

- Suppose predictions are perfect, i.e., $f(x) = f'(x)$ for all $x \in [1..k]$.
  - Assume $k = 10$, $M = 20$

x	1	2	3	4	5	6	7	8	9	10
$f(x) = f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.03	0

  - Profile multiset is $P = \{1^3, 2^{11}, 3^3, 4^2, 6, 7, 9\}$.
    - E.g., there are $\lceil 0.11 \cdot 20 \rceil = 3$ items of size 1 in the profile set.



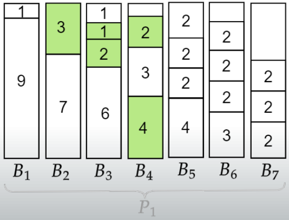
$\sigma = 2, 3, 1, 4$

Profile-Packing Illustration

- Suppose predictions are perfect, i.e., $f(x) = f'(x)$ for all $x \in [1..k]$.
    - Assume $k = 10,\ M = 20$

x	1	2	3	4	5	6	7	8	9	10
$f(x) = f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.03	0

-
    - Profile multiset is $P = \{1^3, 2^{11}, 3^3, 4^2, 6, 7, 9\}$.
        - E.g., there are $\lceil 0.11 \cdot 20 \rceil = 3$ items of size 1 in the profile set.



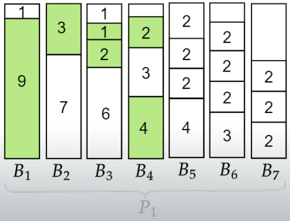
$\sigma = 2, 3, 1, 4, 2$

Profile-Packing Illustration

- Suppose predictions are perfect, i.e., $f(x) = f'(x)$ for all $x \in [1..k]$.
  - Assume $k = 10,\ M = 20$

x	1	2	3	4	5	6	7	8	9	10
$f(x) = f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.03	0

  - Profile multiset is $P = \{1^3, 2^{11}, 3^3, 4^2, 6, 7, 9\}$.
    - E.g., there are $\lceil 0.11 \cdot 20 \rceil = 3$ items of size 1 in the profile set.



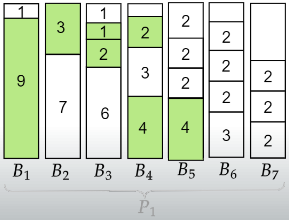
$\sigma = 2, 3, 1, 4, 2, 9$

Profile-Packing Illustration

- Suppose predictions are perfect, i.e., $f(x) = f'(x)$ for all $x \in [1..k]$.
  - Assume $k = 10$, $M = 20$

x	1	2	3	4	5	6	7	8	9	10
$f(x) = f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.03	0

  - Profile multiset is $P = \{1^3, 2^{11}, 3^3, 4^2, 6, 7, 9\}$.
    - E.g., there are $\lceil 0.11 \cdot 20 \rceil = 3$ items of size 1 in the profile set.



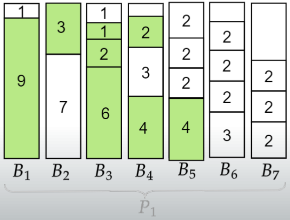
$\sigma = 2, 3, 1, 4, 2, 9, 4$

Profile-Packing Illustration

- Suppose predictions are perfect, i.e., $f(x) = f'(x)$ for all $x \in [1..k]$.
  - Assume $k = 10,\ M = 20$

x	1	2	3	4	5	6	7	8	9	10
$f(x) = f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.03	0

  - Profile multiset is $P = \{1^3, 2^{11}, 3^3, 4^2, 6, 7, 9\}$.
    - E.g., there are $\lceil 0.11 \cdot 20 \rceil = 3$ items of size 1 in the profile set.



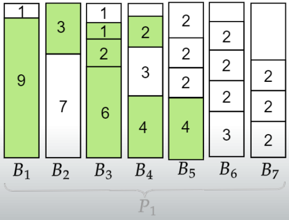
$\sigma = 2,\ 3,\ 1,\ 4,\ 2,\ 9,\ 4,\ 6$

Profile-Packing Illustration

- Suppose predictions are perfect, i.e., $f(x) = f'(x)$ for all $x \in [1..k]$.
  - Assume $k = 10$, $M = 20$

x	1	2	3	4	5	6	7	8	9	10
$f(x) = f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.03	0

  - Profile multiset is $P = \{1^3, 2^{11}, 3^3, 4^2, 6, 7, 9\}$.
    - E.g., there are $\lceil 0.11 \cdot 20 \rceil = 3$ items of size 1 in the profile set.



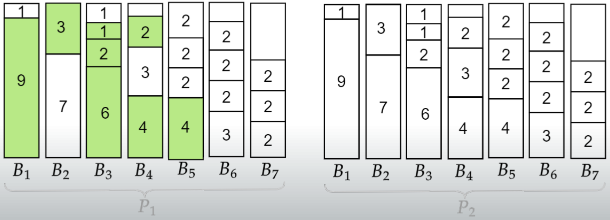
$\sigma = 2, 3, 1, 4, 2, 9, 4, 6, 9$

Profile-Packing Illustration

- Suppose predictions are perfect, i.e., $f(x) = f'(x)$ for all $x \in [1..k]$.
  - Assume $k = 10,\ M = 20$

x	1	2	3	4	5	6	7	8	9	10
$f(x) = f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.03	0

  - Profile multiset is $P = \{1^3, 2^{11}, 3^3, 4^2, 6, 7, 9\}$.
    - E.g., there are $\lceil 0.11 \cdot 20 \rceil = 3$ items of size 1 in the profile set.



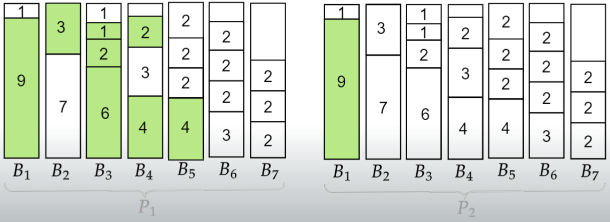
$\sigma = 2, 3, 1, 4, 2, 9, 4, 6, 9$

Profile-Packing Illustration

- Suppose predictions are perfect, i.e., $f(x) = f'(x)$ for all $x \in [1..k]$.
  - Assume $k = 10,\ M = 20$

x	1	2	3	4	5	6	7	8	9	10
$f(x) = f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.03	0

  - Profile multiset is $P = \{1^3, 2^{11}, 3^3, 4^2, 6, 7, 9\}$.
    - E.g., there are $\lceil 0.11 \cdot 20 \rceil = 3$ items of size 1 in the profile set.



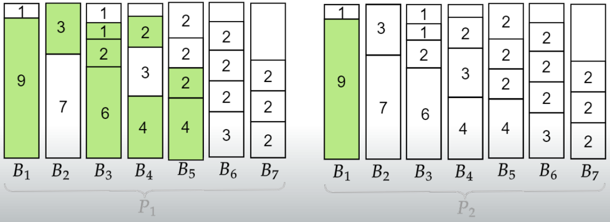
$\sigma = 2,\ 3,\ 1,\ 4,\ 2,\ 9,\ 4,\ 6,\ 9$

Profile-Packing Illustration

- Suppose predictions are perfect, i.e., $f(x) = f'(x)$ for all $x \in [1..k]$.
  - Assume $k = 10$, $M = 20$

x	1	2	3	4	5	6	7	8	9	10
$f(x) = f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.03	0

  - Profile multiset is $P = \{1^3, 2^{11}, 3^3, 4^2, 6, 7, 9\}$.
    - E.g., there are $\lceil 0.11 \cdot 20 \rceil = 3$ items of size 1 in the profile set.



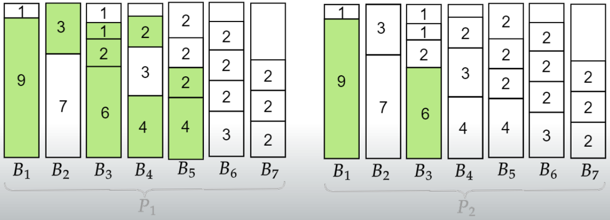
$\sigma = 2, 3, 1, 4, 2, 9, 4, 6, 9, 2$

Profile-Packing Illustration

- Suppose predictions are perfect, i.e., $f(x) = f'(x)$ for all $x \in [1..k]$.
  - Assume $k = 10$, $M = 20$

x	1	2	3	4	5	6	7	8	9	10
$f(x) = f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.03	0

  - Profile multiset is $P = \{1^3, 2^{11}, 3^3, 4^2, 6, 7, 9\}$.
    - E.g., there are $\lceil 0.11 \cdot 20 \rceil = 3$ items of size 1 in the profile set.



$\sigma = 2, 3, 1, 4, 2, 9, 4, 6, 9, 2, 6$

Profile Packing Analysis

Theorem

For any constant $\epsilon \in (0, 0.2]$, and error-free prediction ($f' = f$), Profile-Packing has competitive ratio at most $1 + \epsilon$ [Angelopoulos et al., 2021].

- Profile-Packing has consistency $1 + \epsilon$.

Profile Packing Analysis

Theorem

For any constant $\epsilon \in (0, 0.2]$, and error-free prediction ($f' = f$), Profile-Packing has competitive ratio at most $1 + \epsilon$ [Angelopoulos et al., 2021].

- Profile-Packing has consistency $1 + \epsilon$.
- What about noisy predictions?

Profile-Packing with Noisy Predictions

- Define the error as the L_1 distance between vectors f and f'.

x	1	2	3	4	5	6	7	8	9	10
$f(x)$	0.10	0.53	0.15	0.10	0.05	0.03	0.1	0	0.05	0
$f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.05	0

$$\eta = |0.10 - 0.11| + |0.05 - 0| + |0.1 - 0.05| = 0.2$$

Profile-Packing with Noisy Predictions

- Define the error as the L_1 distance between vectors f and f'.

x	1	2	3	4	5	6	7	8	9	10
$f(x)$	0.10	0.53	0.15	0.10	0.05	0.03	0.1	0	0.05	0
$f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.05	0

$$\eta = |0.10 - 0.11| + |0.05 - 0| + |0.1 - 0.05| = 0.2$$

- The algorithm works as before but some placeholders remain empty.
  - **Special items** with predicted frequency 0 are treated using First-Fit.

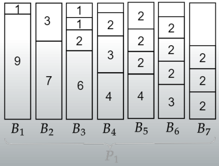
Profile-Packing with Noisy Predictions

- Define the error as the L_1 distance between vectors f and f'.

x	1	2	3	4	5	6	7	8	9	10
$f(x)$	0.10	0.53	0.15	0.10	0.05	0.03	0.1	0	0.05	0
$f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.05	0

$$\eta = |0.10 - 0.11| + |0.05 - 0| + |0.1 - 0.05| = 0.2$$

- The algorithm works as before but some placeholders remain empty.
  - **Special items** with predicted frequency 0 are treated using First-Fit.



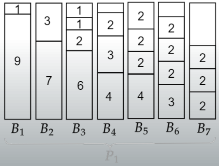
Profile-Packing with Noisy Predictions

- Define the error as the L_1 distance between vectors f and f'.

x	1	2	3	4	5	6	7	8	9	10
$f(x)$	0.10	0.53	0.15	0.10	0.05	0.03	0.1	0	0.05	0
$f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.05	0

$$\eta = |0.10 - 0.11| + |0.05 - 0| + |0.1 - 0.05| = 0.2$$

- The algorithm works as before but some placeholders remain empty.
  - **Special items** with predicted frequency 0 are treated using First-Fit.



$\sigma = 2$

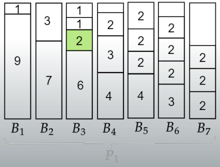
Profile-Packing with Noisy Predictions

- Define the error as the L_1 distance between vectors f and f'.

x	1	2	3	4	5	6	7	8	9	10
$f(x)$	0.10	0.53	0.15	0.10	0.05	0.03	0.1	0	0.05	0
$f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.05	0

$$\eta = |0.10 - 0.11| + |0.05 - 0| + |0.1 - 0.05| = 0.2$$

- The algorithm works as before but some placeholders remain empty.
    - **Special items** with predicted frequency 0 are treated using First-Fit.



$\sigma = 2$

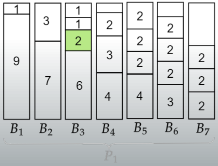
Profile-Packing with Noisy Predictions

- Define the error as the L_1 distance between vectors f and f'.

x	1	2	3	4	5	6	7	8	9	10
$f(x)$	0.10	0.53	0.15	0.10	0.05	0.03	0.1	0	0.05	0
$f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.05	0

$$\eta = |0.10 - 0.11| + |0.05 - 0| + |0.1 - 0.05| = 0.2$$

- The algorithm works as before but some placeholders remain empty.
    - **Special items** with predicted frequency 0 are treated using First-Fit.



σ = 2, 3

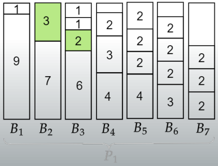
Profile-Packing with Noisy Predictions

- Define the error as the L_1 distance between vectors f and f'.

x	1	2	3	4	5	6	7	8	9	10
$f(x)$	0.10	0.53	0.15	0.10	0.05	0.03	0.1	0	0.05	0
$f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.05	0

$$\eta = |0.10 - 0.11| + |0.05 - 0| + |0.1 - 0.05| = 0.2$$

- The algorithm works as before but some placeholders remain empty.
  - **Special items** with predicted frequency 0 are treated using First-Fit.



$\sigma = 2, 3$

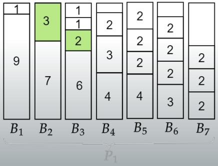
Profile-Packing with Noisy Predictions

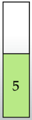
- Define the error as the L_1 distance between vectors f and f'.

x	1	2	3	4	5	6	7	8	9	10
$f(x)$	0.10	0.53	0.15	0.10	0.05	0.03	0.1	0	0.05	0
$f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.05	0

$$\eta = |0.10 - 0.11| + |0.05 - 0| + |0.1 - 0.05| = 0.2$$

- The algorithm works as before but some placeholders remain empty.
  - **Special items** with predicted frequency 0 are treated using First-Fit.





$\sigma = 2, 3, 5$

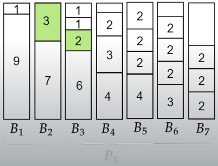
Profile-Packing with Noisy Predictions

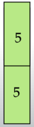
- Define the error as the L_1 distance between vectors f and f'.

x	1	2	3	4	5	6	7	8	9	10
$f(x)$	0.10	0.53	0.15	0.10	0.05	0.03	0.1	0	0.05	0
$f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.05	0

$$\eta = |0.10 - 0.11| + |0.05 - 0| + |0.1 - 0.05| = 0.2$$

- The algorithm works as before but some placeholders remain empty.
  - **Special items** with predicted frequency 0 are treated using First-Fit.





σ = 2, 3, 5,5

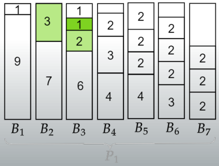
Profile-Packing with Noisy Predictions

- Define the error as the L_1 distance between vectors f and f'.

x	1	2	3	4	5	6	7	8	9	10
$f(x)$	0.10	0.53	0.15	0.10	0.05	0.03	0.1	0	0.05	0
$f'(x)$	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.05	0

$$\eta = |0.10 - 0.11| + |0.05 - 0| + |0.1 - 0.05| = 0.2$$

- The algorithm works as before but some placeholders remain empty.
  - **Special items** with predicted frequency 0 are treated using First-Fit.



$\sigma = 2,\ 3,\ 5, 5,\ 1$

Profile-Packing with Noisy Predictions

Theorem

For any constant $\epsilon \in (0, 0.2]$*, and predictions* f' *with error* η*, Profile-Packing has competitive ratio at most* $1 + (2 + 5\epsilon)\eta k + \epsilon$*.*

- Consistency is $1 + \epsilon$ (when $\eta = 0$).
- Robustness grows with k:
  - Can we improve over this?

Consistency-Robustness Trade off

Theorem

Any $(1+\epsilon)$-consistent algorithm has robustness that grows with k.

Consistency-Robustness Trade off

Theorem

Any $(1+\epsilon)$-consistent algorithm has robustness that grows with k.

- Consider input sequences that start with a common prefix formed by n items of size 1.

Consistency-Robustness Trade off

Theorem

Any $(1+\epsilon)$-consistent algorithm has robustness that grows with k.

- Consider input sequences that start with a common prefix formed by n items of size 1.
  - Suppose predictions indicate that half of items are of size 1 and half are of size $k-1$.

Consistency-Robustness Trade off

Theorem

Any $(1+\epsilon)$-consistent algorithm has robustness that grows with k.

- Consider input sequences that start with a common prefix formed by n items of size 1.
  - Suppose predictions indicate that half of items are of size 1 and half are of size $k-1$.
  - $\sigma_1 = \underbrace{1, 1, \ldots, 1}_{n \text{ items}}, \underbrace{k-1, k-1, \ldots, k-1}_{n \text{ items}} \quad (\eta = 0)$
    - To be $(1+\epsilon)$-consistent, the algorithm should open at least $(1-k\epsilon)n$ bins for the first n items

Consistency-Robustness Trade off

Theorem

Any $(1+\epsilon)$-consistent algorithm has robustness that grows with k.

- Consider input sequences that start with a common prefix formed by n items of size 1.
  - Suppose predictions indicate that half of items are of size 1 and half are of size $k-1$.
  - $\sigma_1 = \underbrace{1, 1, \ldots, 1}_{n \text{ items}}, \underbrace{k-1, k-1, \ldots, k-1}_{n \text{ items}} \quad (\eta = 0)$
    - To be $(1+\epsilon)$-consistent, the algorithm should open at least $(1-k\epsilon)n$ bins for the first n items
  - $\sigma_2 = \underbrace{1, 1, \ldots, 1}_{n \text{ items}}, \underbrace{1, 1, \ldots, 1}_{n \text{ items}} \quad (\eta = 1)$
    - Given that the algorithm opens at least $(1-k\epsilon)n$, robustness is at least $(1-c)k$, assuming $\epsilon \leq c/k$.

A Hybrid Algorithm

- A Hybrid$_A(\lambda)$ algorithm between Profile-Packing and A
  - A can be any online algorithm (e.g., First-Fit or Super-Harmonic).

A Hybrid Algorithm

- A Hybrid$_A(\lambda)$ algorithm between Profile-Packing and A
  - A can be any online algorithm (e.g., First-Fit or Super-Harmonic).
  - Each item x is declared as either PP-item or A-item.
  - For each $x \in [1..k]$, there are counters `ppcount(x)` and `count(x)`

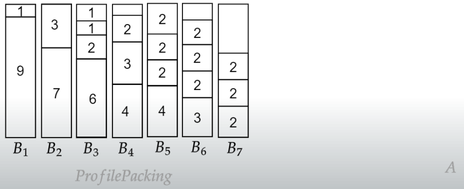
A Hybrid Algorithm

- A Hybrid$_A(\lambda)$ algorithm between Profile-Packing and A
  - A can be any online algorithm (e.g., First-Fit or Super-Harmonic).
  - Each item x is declared as either PP-item or A-item.
  - For each $x \in [1..k]$, there are counters $\texttt{ppcount(x)}$ and $\texttt{count(x)}$
    - If there is a place-holder of size x in a non-empty bin, then x is declared a pp-item and placed in the placeholder.
    - Otherwise, x is declared as a pp-item iff $\texttt{ppcount(x)} \leq \lambda\texttt{count(x)}$ and an A-item otherwise.

A Hybrid Algorithm

- A Hybrid$_A(\lambda)$ algorithm between Profile-Packing and A
  - A can be any online algorithm (e.g., First-Fit or Super-Harmonic).
  - Each item x is declared as either PP-item or A-item.
  - For each $x \in [1..k]$, there are counters `ppcount(x)` and `count(x)`
    - If there is a place-holder of size x in a non-empty bin, then x is declared a pp-item and placed in the placeholder.
    - Otherwise, x is declared as a pp-item iff $\texttt{ppcount(x)} \le \lambda\texttt{count(x)}$ and an A-item otherwise.

Assume $\lambda = 2/3$.



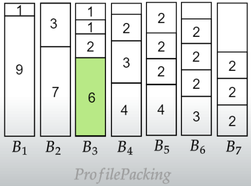
$\sigma = 6$

ppcount(6) = 0
count(6) = 0

A Hybrid Algorithm

- A Hybrid$_A(\lambda)$ algorithm between Profile-Packing and A
  - A can be any online algorithm (e.g., First-Fit or Super-Harmonic).
  - Each item x is declared as either PP-item or A-item.
  - For each $x \in [1..k]$, there are counters `ppcount(x)` and `count(x)`
    - If there is a place-holder of size x in a non-empty bin, then x is declared a pp-item and placed in the placeholder.
    - Otherwise, x is declared as a pp-item iff `ppcount(x)` $\leq \lambda$`count(x)` and an A-item otherwise.

Assume $\lambda = 2/3$.



A

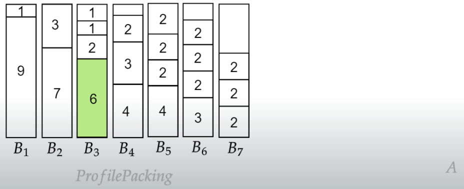
$\sigma = 6$

ppcount(6) = 1
count(6) = 1

A Hybrid Algorithm

- A Hybrid$_A(\lambda)$ algorithm between Profile-Packing and A
  - A can be any online algorithm (e.g., First-Fit or Super-Harmonic).
  - Each item x is declared as either PP-item or A-item.
  - For each $x \in [1..k]$, there are counters ppcount(x) and count(x)
    - If there is a place-holder of size x in a non-empty bin, then x is declared a pp-item and placed in the placeholder.
    - Otherwise, x is declared as a pp-item iff ppcount(x) $\leq \lambda$count(x) and an A-item otherwise.

Assume $\lambda = 2/3$.



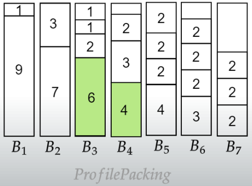
$\sigma = 6, 4$

ppcount(4) = 0
count(4) = 0

A Hybrid Algorithm

- A Hybrid$_A(\lambda)$ algorithm between Profile-Packing and A
  - A can be any online algorithm (e.g., First-Fit or Super-Harmonic).
  - Each item x is declared as either PP-item or A-item.
  - For each $x \in [1..k]$, there are counters `ppcount(x)` and `count(x)`
    - If there is a place-holder of size x in a non-empty bin, then x is declared a pp-item and placed in the placeholder.
    - Otherwise, x is declared as a pp-item iff $\texttt{ppcount(x)} \leq \lambda\texttt{count(x)}$ and an A-item otherwise.

Assume $\lambda = 2/3$.



A

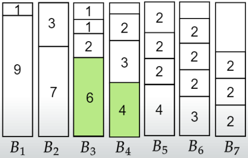
$\sigma = 6, 4$

ppcount(4) = 1
count(4) = 1

A Hybrid Algorithm

- A Hybrid$_A(\lambda)$ algorithm between Profile-Packing and A
  - A can be any online algorithm (e.g., First-Fit or Super-Harmonic).
  - Each item x is declared as either PP-item or A-item.
  - For each $x \in [1..k]$, there are counters `ppcount(x)` and `count(x)`
    - If there is a place-holder of size x in a non-empty bin, then x is declared a pp-item and placed in the placeholder.
    - Otherwise, x is declared as a pp-item iff $\texttt{ppcount(x)} \leq \lambda\texttt{count(x)}$ and an A-item otherwise.

Assume $\lambda = 2/3$.



ProfilePacking

A

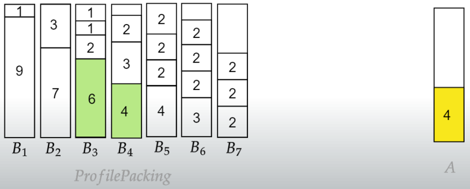
$\sigma = 6, 4, 4$

ppcount(4) = 1
count(4) = 1

A Hybrid Algorithm

- A Hybrid$_A(\lambda)$ algorithm between Profile-Packing and A
  - A can be any online algorithm (e.g., First-Fit or Super-Harmonic).
  - Each item x is declared as either PP-item or A-item.
  - For each $x \in [1..k]$, there are counters `ppcount(x)` and `count(x)`
    - If there is a place-holder of size x in a non-empty bin, then x is declared a pp-item and placed in the placeholder.
    - Otherwise, x is declared as a pp-item iff $\texttt{ppcount(x)} \leq \lambda\texttt{count(x)}$ and an A-item otherwise.

Assume $\lambda = 2/3$.



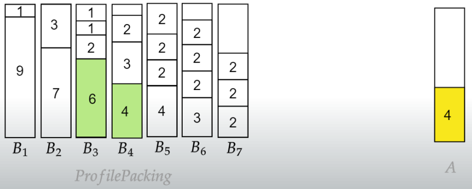
$\sigma = 6, 4, 4$

ppcount(4) = 1
count(4) = 2

A Hybrid Algorithm

- A Hybrid$_A(\lambda)$ algorithm between Profile-Packing and A
  - A can be any online algorithm (e.g., First-Fit or Super-Harmonic).
  - Each item x is declared as either PP-item or A-item.
  - For each $x \in [1..k]$, there are counters `ppcount(x)` and `count(x)`
    - If there is a place-holder of size x in a non-empty bin, then x is declared a pp-item and placed in the placeholder.
    - Otherwise, x is declared as a pp-item iff $\texttt{ppcount(x)} \leq \lambda\texttt{count(x)}$ and an A-item otherwise.

Assume $\lambda = 2/3$.



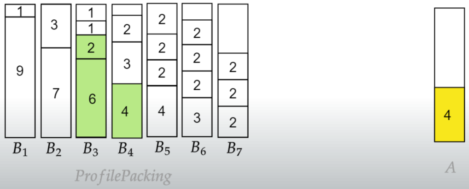
$\sigma = 6, 4, 4, 2$

ppcount(2) = 0
count(2) = 0

A Hybrid Algorithm

- A Hybrid$_A(\lambda)$ algorithm between Profile-Packing and A
  - A can be any online algorithm (e.g., First-Fit or Super-Harmonic).
  - Each item x is declared as either PP-item or A-item.
  - For each $x \in [1..k]$, there are counters `ppcount(x)` and `count(x)`
    - If there is a place-holder of size x in a non-empty bin, then x is declared a pp-item and placed in the placeholder.
    - Otherwise, x is declared as a pp-item iff $\texttt{ppcount(x)} \leq \lambda\texttt{count(x)}$ and an A-item otherwise.

Assume $\lambda = 2/3$.



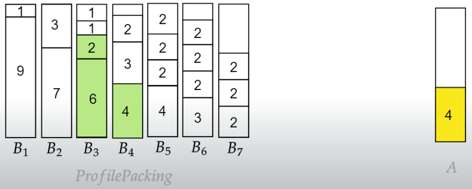
$\sigma = 6, 4, 4, 2$

ppcount(2) = 1
count(2) = 1

A Hybrid Algorithm

- A Hybrid$_A(\lambda)$ algorithm between Profile-Packing and A
  - A can be any online algorithm (e.g., First-Fit or Super-Harmonic).
  - Each item x is declared as either PP-item or A-item.
  - For each $x \in [1..k]$, there are counters `ppcount(x)` and `count(x)`
    - If there is a place-holder of size x in a non-empty bin, then x is declared a pp-item and placed in the placeholder.
    - Otherwise, x is declared as a pp-item iff $\texttt{ppcount(x)} \leq \lambda\texttt{count(x)}$ and an A-item otherwise.

Assume $\lambda = 2/3$.



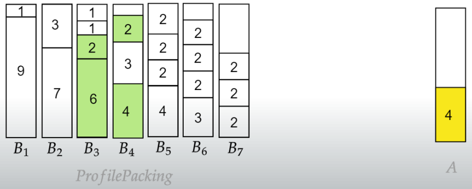
$\sigma = 6, 4, 4, 2, 2$

ppcount(2) = 1
count(2) = 1

A Hybrid Algorithm

- A Hybrid$_A(\lambda)$ algorithm between Profile-Packing and A
  - A can be any online algorithm (e.g., First-Fit or Super-Harmonic).
  - Each item x is declared as either PP-item or A-item.
  - For each $x \in [1..k]$, there are counters `ppcount(x)` and `count(x)`
    - If there is a place-holder of size x in a non-empty bin, then x is declared a pp-item and placed in the placeholder.
    - Otherwise, x is declared as a pp-item iff $\texttt{ppcount(x)} \leq \lambda\texttt{count(x)}$ and an A-item otherwise.

Assume $\lambda = 2/3$.



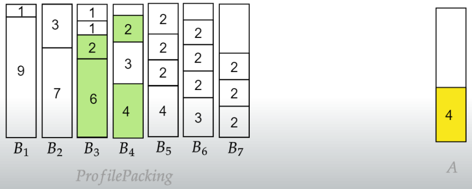
$\sigma = 6, 4, 4, 2, 2$

ppcount(2) = 2
count(2) = 2

A Hybrid Algorithm

- A Hybrid$_A(\lambda)$ algorithm between Profile-Packing and A
  - A can be any online algorithm (e.g., First-Fit or Super-Harmonic).
  - Each item x is declared as either PP-item or A-item.
  - For each $x \in [1..k]$, there are counters `ppcount(x)` and `count(x)`
    - If there is a place-holder of size x in a non-empty bin, then x is declared a pp-item and placed in the placeholder.
    - Otherwise, x is declared as a pp-item iff $\texttt{ppcount(x)} \leq \lambda\texttt{count(x)}$ and an A-item otherwise.

Assume $\lambda = 2/3$.



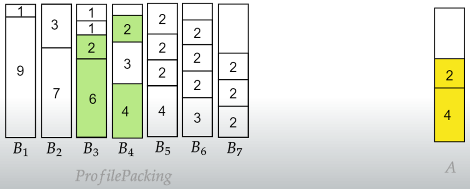
$\sigma = 6, 4, 4, 2, 2, 2$

ppcount(2) = 2
count(2) = 2

A Hybrid Algorithm

- A Hybrid$_A(\lambda)$ algorithm between Profile-Packing and A
  - A can be any online algorithm (e.g., First-Fit or Super-Harmonic).
  - Each item x is declared as either PP-item or A-item.
  - For each $x \in [1..k]$, there are counters `ppcount(x)` and `count(x)`
    - If there is a place-holder of size x in a non-empty bin, then x is declared a pp-item and placed in the placeholder.
    - Otherwise, x is declared as a pp-item iff `ppcount(x)` $\leq \lambda$`count(x)` and an A-item otherwise.

Assume $\lambda = 2/3$.



$\sigma = 6, 4, 4, 2, 2, 2$

ppcount(2) = 2
count(2) = 3

Analysis of Hybrid

Theorem

For any $\epsilon \in (0, 0.2]$ *and* $\lambda \in [0, 1]$, $\textsc{Hybrid}(\lambda)$ *has competitive ratio* $(1+\epsilon)((1+(2+5\epsilon)\eta k+\epsilon)\lambda + c_A(1-\lambda))$, *where* c_A *is the competitive ratio of A.*

Analysis of Hybrid

Theorem

For any $\epsilon \in (0, 0.2]$ and $\lambda \in [0, 1]$, $\textsc{Hybrid}(\lambda)$ has competitive ratio $(1+\epsilon)((1+(2+5\epsilon)\eta k+\epsilon)\lambda + c_A(1-\lambda))$, where c_A is the competitive ratio of A.

Corollary

For any $\epsilon \in (0, 0.2]$ and $\lambda \in [0, 1]$, there is an algorithm with competitive ratio $(1+\epsilon)(1.5783 + \lambda((2+5\epsilon)\eta k - 0.5783 + \epsilon))$.

Experimental Results

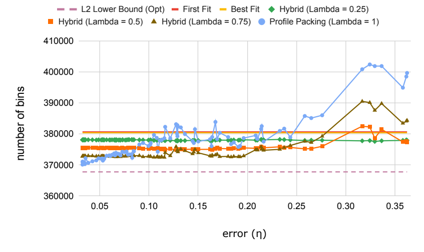
- We study the typical performance of Profile-Packing and $\textsc{Hybrid}(\lambda)$ using experiments.

Experimental Results

- We study the typical performance of Profile-Packing and $\textsc{Hybrid}(\lambda)$ using experiments.
- Create sequences using Weibull distribution or from the BIBLib bin packing library.
    - We have $n = 10^6$, $M = 5000$, $k = 100$, and use FFD for packing profile.
    - Predictions are defined based on frequencies in prefixes of different lengths.

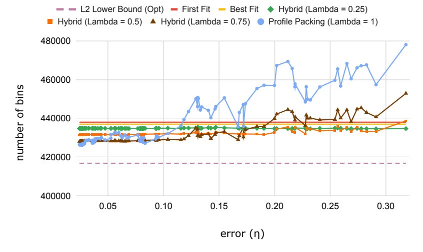
Experimental Results

- We study the typical performance of Profile-Packing and $\textsc{Hybrid}(\lambda)$ using experiments.
- Create sequences using Weibull distribution or from the BIBLib bin packing library.
  - We have $n = 10^6$, $M = 5000$, $k = 100$, and use FFD for packing profile.
  - Predictions are defined based on frequencies in prefixes of different lengths.



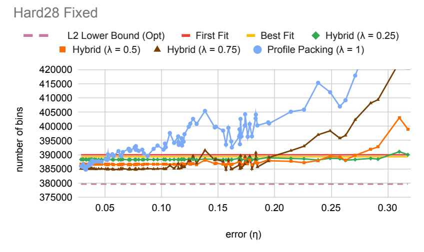
Experimental Results

- We study the typical performance of Profile-Packing and $\textsc{Hybrid}(\lambda)$ using experiments.
- Create sequences using Weibull distribution or from the BIBLib bin packing library.
    - We have $n = 10^6$, $M = 5000$, $k = 100$, and use FFD for packing profile.
    - Predictions are defined based on frequencies in prefixes of different lengths.



Experimental Results

- We study the typical performance of Profile-Packing and $\textsc{Hybrid}(\lambda)$ using experiments.
- Create sequences using Weibull distribution or from the BIBLib bin packing library.
  - We have $n = 10^6$, $M = 5000$, $k = 100$, and use FFD for packing profile.
  - Predictions are defined based on frequencies in prefixes of different lengths.



Conclusions

https://www.cnbc.com/2020/04/10/
coronavirus-empty-streets-around-the-world-are-attracting-wildlife.html

Concluding Remarks

- Most current results consider **static predictions**.
    - First the predictions are generated and then the input is revealed.

Concluding Remarks

- Most current results consider **static predictions**.
    - First the predictions are generated and then the input is revealed.
    - It is possible to update predictions in the course of the algorithm.

Concluding Remarks

- Most current results consider **static predictions**.
  - First the predictions are generated and then the input is revealed.
  - It is possible to update predictions in the course of the algorithm.
  - **Adaptive algorithm:** maintain frequencies in a window of size w formed by the last w items.

Concluding Remarks

- Predictions about frequencies can be helpful for improving online algorithms.

Concluding Remarks

- Predictions about frequencies can be helpful for improving online algorithms.
- One can use other error measures, e.g., Earth-Mover-Distance.

Concluding Remarks

- Predictions about frequencies can be helpful for improving online algorithms.
- One can use other error measures, e.g., Earth-Mover-Distance.
- Not only the competitive ratio, but more importantly the typical performance of algorithms can be improved, using predictions.

References

Angelopoulos, S.; Dürr, C.; Jin, S.; Kamali, S.; and Renault, M. P. (2020).
"Online Computation with Untrusted Advice".
In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 52:1–52:15.

Angelopoulos, S.; Dürr, C.; Kamali, S.; Renault, M. P.; and Rosén, A. (2018).
"Online Bin Packing with Advice of Small Size".
Theory Computing Systems, 62(8), pp. 2006–2034.

Angelopoulos, S. and Kamali, S. (2021).
"Contract Scheduling With Predictions".
In *Proceedings of AAAI.*

Angelopoulos, S.; Kamali, S.; and Shadkami, K. (2021).
"Online Bin Packing with Predictions".
CoRR, abs/2102.03311.

Antoniadis, A.; Coester, C.; Eliás, M.; Polak, A.; and Simon, B. (2020).
"Online metric algorithms with untrusted predictions".
In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 345–355.

Balogh, J.; Békési, J.; Dósa, G.; Epstein, L.; and Levin, A. (2018).
"A New and Improved Algorithm for Online Bin Packing".
In *Proceedings of the 26th European Symposium on Algorithms (ESA)*, volume 112, pages 5:1–5:14.

Balogh, J.; Békési, J.; Dósa, G.; Epstein, L.; and Levin, A. (2019).
"A New Lower Bound for Classic Online Bin Packing".
In *Proceedings of the 17th Workshop on Approximation and Online Algorithms (WAOA)*, pages 18–28.

Boyar, J.; Kamali, S.; Larsen, K. S.; and López-Ortiz, A. (2016).
"Online Bin Packing with Advice".
Algorithmica, 74(1), pp. 507–527.

Delorme, M.; Iori, M.; and Martello, S.
"BPPLIB–A Bin Packing Problem Library".
Accessed: 2021-01-15.

Lattanzi, S.; Lavastida, T.; Moseley, B.; and Vassilvitskii, S. (2020).
"Online scheduling via learned weights".
In *Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1859–1877.

Lykouris, T. and Vassilvitskii, S. (2018).
"Competitive Caching with Machine Learned Advice".
In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 3302–3311.

Mikkelsen, J. W. (2016).
"Randomization Can Be as Helpful as a Glimpse of the Future in Online Computation".
In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 55, pages 39:1–39:14.

Purohit, M.; Svitkina, Z.; and Kumar, R. (2018).
"Improving Online Algorithms via ML Predictions".
In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS)*, volume 31, pages 9661–9670.

Renault, M. P.; Rosén, A.; and van Stee, R. (2015).
"Online algorithms with advice for bin packing and scheduling problems".
Theor. Comput. Sci., 600, pp. 155–170.

Rohatgi, D. (2020).
"Near-optimal bounds for online caching with machine learned advice".
In *Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1834–1845.