Analysis of the Harmonic-k Algorithm for One-Dimensional Bin Packing

Shreyas Ramasubramanian

1 Introduction

The one-dimensional bin packing problem (1DBPP) is a classic NP-hard combinatorial optimization problem. The goal is to pack a set of items, $I = \{i_1, i_2, \ldots, i_n\}$, with sizes $s(i) \in (0, 1]$, into the minimum number of unit-capacity bins.

Given its NP-hard nature, polynomial-time approximation algorithms are necessary for practical solutions. These algorithms are often classified as **offline**, where all items are known in advance (like First-Fit Decreasing), or **online**, where items arrive one by one and must be packed without knowledge of future items.

This report details the **Harmonic-k (Hk) algorithm**, which your project document identifies as an "advanced, structured online heuristic". It represents a significant step up from simple online heuristics like First-Fit (FF) by using a more structured grouping strategy to achieve a better performance guarantee.

2 The Harmonic-k (Hk) Algorithm

2.1 Core Strategy

The key idea behind the Harmonic-k algorithm is to partition items by size and pack them into dedicated bins for each size class. It is parameterized by an integer $k \geq 1$.

The algorithm partitions the item size interval $(0, 1]$ into k disjoint sub-intervals, $I_1, \ldots, I_k$. A common definition for this "harmonic partition" is:

- $I_1 = (1/2, 1]$
- $I_2 = (1/3, 1/2]$
- $I_3 = (1/4, 1/3]$
- $\ldots$
- $I_{k-1} = (1/k, 1/(k-1)]$
- $I_k = (0, 1/k]$

Items belonging to $I_1, \ldots, I_{k-1}$ are considered **"large items"**. Items in I_k are considered **"small items"**.

The algorithm maintains k separate groups of bins.

- $k-1$ groups of bins, $B_1, \ldots, B_{k-1}$, are reserved for the "large" items. Bins in B_j will *only* ever contain items from I_j.
- One group of bins, B_k, is reserved for all the "small" items from I_k.

2.2 Algorithm Pseudocode

The algorithm processes each item j as it arrives:

1. **Receive Item:** Let the new item be j with size s_j.
2. **Classify Item:** Find the unique interval I_i such that $s_j \in I_i$.
3. **Pack Item:**
   - **If $i < k$ (Item is "large"):**
     - Apply the **First-Fit (FF)** heuristic to item j, using *only* the bins in the dedicated group B_i.
     - That is, place j in the first bin in B_i that has enough space.
     - If no bin in B_i has space, open a new bin, add it to group B_i, and place j inside it.
   - **If $i = k$ (Item is "small"):**
     - Apply the **First-Fit (FF)** heuristic to item j, using *only* the bins in the "small item" group B_k.
     - Place j in the first bin in B_k that has space.
     - If no bin in B_k has space, open a new bin, add it to group B_k, and place j inside it.

3 Analysis of the Algorithm

3.1 Time Complexity

Let n be the total number of items.

1. **Classification:** For each item, finding its interval I_i can be done in $O(\log k)$ time with a binary search on the interval boundaries $\{1/2, 1/3, \ldots, 1/k\}$, or $O(k)$ time with a linear scan. Since k is a fixed constant, this is $O(1)$.
2. **Packing:** The packing step involves a First-Fit (FF) operation. The standard time complexity for n items using FF is $O(n \log n)$ (by using a balanced binary tree to manage the bin-fill levels).
3. **Overall:** Since the n items are partitioned among k groups, and the total number of FF operations is n, the total time complexity is dominated by the First-Fit placements, resulting in $O(n \log n)$.

3.2 Approximation Ratio (Weight Function Proof)

To prove the asymptotic approximation ratio, we use the **Weight Function Method**. The goal is to define a weight $w(i)$ for each item i based on its size $s(i)$.

3.2.1 The Strategy

Let $W = \sum w(i)$ be the total weight of all items. The proof rests on finding two constants, L and U:

1. **Lower Bound (L):** Prove that every bin closed by the Harmonic-k algorithm (Hk) contains items with a total weight of at least L.

2. **Upper Bound (U):** Prove that *any* bin (including one in an optimal packing) can contain items with a total weight of at most U.

If we establish L and U, we can bound the number of bins. Let $Hk(I)$ be the bins used by the algorithm and $OPT(I)$ be the optimal number of bins.

- $Hk(I)$ uses Hk_{closed} closed bins and Hk_{open} open bins. By definition of Hk, $Hk_{open} \leq k$.
- Total weight $W \geq L \cdot Hk_{closed}$. This implies $Hk_{closed} \leq W/L$.
- Therefore, $Hk(I) = Hk_{closed} + Hk_{open} \leq \frac{W}{L} + k$.
- For the optimal solution, $W \leq U \cdot OPT(I)$, because each of the $OPT(I)$ bins can hold at most U weight. This implies $OPT(I) \geq \frac{W}{U}$.

The approximation ratio is $\frac{Hk(I)}{OPT(I)} \leq \frac{W/L+k}{W/U} = \frac{U}{L} + \frac{k \cdot U}{W}$.

For the *asymptotic* ratio, we let $OPT(I) \to \infty$. This implies $W \to \infty$. The second term, $\frac{k \cdot U}{W}$, thus goes to 0.

$$R_{Hk} = \lim_{OPT \to \infty} \frac{Hk(I)}{OPT(I)} \leq \frac{U}{L}$$

Our goal is to define weights $w(i)$ that minimize this U/L ratio.

3.2.2 Defining the Weights

We define the weights $w(i)$ for an item i with size $s(i)$ as follows:

- If $s(i) \in I_1 = (1/2, 1]$, $w(i) = 1$.
- If $s(i) \in I_2 = (1/3, 1/2]$, $w(i) = 1/2$.
- If $s(i) \in I_3 = (1/4, 1/3]$, $w(i) = 1/3$.
- ...
- If $s(i) \in I_{k-1} = (1/k, 1/(k-1)]$, $w(i) = 1/(k-1)$.
- If $s(i) \in I_k = (0, 1/k]$, $w(i) = \frac{k}{k-1} \cdot s(i)$.

3.2.3 Part 1: Finding the Lower Bound (L)

We prove that any bin closed by the Hk algorithm has a total weight of at least 1. Thus, $\mathbf{L = 1}$. We check this for each category of bin B_j that gets closed.

Case 1: Bins B_1 (for $j = 1$). Items i have $s(i) > 1/2$. Only one such item fits in a bin. When this bin is "closed" (i.e., it contains its single item), its total weight is $w(i) = 1$.

Case 2: Bins B_j (for $1 < j < k$). Items i have $s(i) \in (1/(j+1), 1/j]$. A bin can hold at most j such items (since $(j+1) \cdot s(i) > 1$). The Hk algorithm uses First-Fit and closes a bin when the next item does not fit. This means a closed bin B_j must contain exactly j items. The weight of each item is $w(i) = 1/j$. Total weight $= j \cdot (1/j) = 1$.

Case 3: Bins B_k (for $j = k$). Items i have $s(i) \in (0, 1/k]$. A bin B_k is closed when an arriving item i_{next} does not fit. This means the empty space in the bin is $< s(i_{next}) \leq 1/k$. Let S be the total size of items already in the bin. $S > 1 - s(i_{next}) \geq 1 - 1/k$. So, $S > \frac{k-1}{k}$. The total weight W_k in the bin is the sum of the weights of its items: $W_k = \sum w(i) = \sum \left(\frac{k}{k-1} \cdot s(i) \right)$ $W_k = \frac{k}{k-1} \cdot \sum s(i) = \frac{k}{k-1} \cdot S$ Since $S > \frac{k-1}{k}$, we have $W_k > \frac{k}{k-1} \cdot \frac{k-1}{k} = 1$.

In all cases, a closed bin has a total weight of at least 1. Therefore, $\mathbf{L = 1}$.

3.2.4 Part 2: Bounding the Upper Bound (U)

We must find the maximum possible weight U that can fit into *any* single bin. We do this by finding the "densest" possible packing. First, let's analyze the density $\rho(i) = w(i)/s(i)$ for each interval:

- I_1: $s > 1/2, w = 1 \implies \rho(i) = 1/s < 1/(1/2) = 2$.
- I_2: $s > 1/3, w = 1/2 \implies \rho(i) = (1/2)/s < (1/2)/(1/3) = 3/2 = 1.5$.
- I_3: $s > 1/4, w = 1/3 \implies \rho(i) = (1/3)/s < (1/3)/(1/4) = 4/3 \approx 1.33$.
- I_k: $s \leq 1/k, w = \frac{k}{k-1}s \implies \rho(i) = \frac{k}{k-1}$. (As $k \to \infty$, $\rho \to 1$).

The "densest" items are those in I_1. We analyze the worst-case bin via case analysis.

Case 1: The bin contains no I_1 items. The maximum density of any item is $\rho_{max} < 3/2$ (from I_2). The total weight $W = \sum w(i) = \sum \rho(i)s(i) \leq \rho_{max} \cdot \sum s(i)$. Since $\sum s(i) \leq 1$ (bin capacity), $W < (3/2) \cdot 1 = 1.5$.

Case 2: The bin contains one I_1 item. Let this item be i_1. Its weight is $w(i_1) = 1$. Its size is $s(i_1) > 1/2$. The remaining space is $S_{rem} = 1 - s(i_1) < 1/2$.

- **Subcase 2a: No I_2 items.** The next densest items are from I_3 ($\rho < 4/3$). $W_{rem} < (4/3) \cdot S_{rem} < (4/3) \cdot (1/2) = 2/3$. Total $W = w(i_1) + W_{rem} < 1 + 2/3 = 5/3 \approx 1.667$.
- **Subcase 2b: One I_2 item.** Let this be i_2. Its weight is $w(i_2) = 1/2$. Its size is $s(i_2) > 1/3$. (A second I_2 item cannot fit, as $2 \cdot (1/3) > 1/2$). Total weight so far is $1 + 1/2 = 1.5$. Remaining space $S_{rem} < 1 - s(i_1) - s(i_2) < 1 - (1/2) - (1/3) = 1/6$.
- **Subcase 2c (Iterative continuation):** We now fill the remaining $1/6$ space. Items from $I_3(s > 1/4)$, $I_4(s > 1/5)$, $I_5(s > 1/6)$ cannot fit. The next densest items that **can** fit are from I_6 (where $s > 1/7$ and $w = 1/6$). As shown in the video analysis, this process can be iterated: one I_1 item, one I_2 item, one I_6 item, one I_{42} item, and so on. This constructs the worst-case, densest bin.

  This iterative analysis shows that the total weight W converges to a sum:

$$W_{max} = \sum_{j=1}^{\infty} w(i_j) \text{ where } i_j \in I_{r_j}$$

  This sum is known to converge to $\mathbf{U \approx 1.69103\ldots}$

The maximum weight found across all cases is $U \approx 1.69103$.

3.2.5 Conclusion: The Asymptotic Ratio

We have established:

- $L = 1$ (Lower bound on closed Hk bin weight)
- $U \approx 1.69103\ldots$ (Upper bound on any bin's weight)

The asymptotic ratio R_{Hk} is:

$$R_{Hk} \leq \frac{U}{L} \approx \frac{1.69103\ldots}{1} \approx 1.69103\ldots$$

This proves that for a sufficiently large k, the Harmonic-k algorithm achieves an asymptotic performance guarantee of ≈ 1.691, which is superior to the 1.7 ratio of the standard First-Fit algorithm.

4 Conclusion

The Harmonic-k algorithm serves as a key example of a structured online heuristic, as noted in the project proposal. It achieves a better performance guarantee than simple heuristics by accepting a slightly more complex implementation (managing k groups of bins). It effectively bridges the gap between simple, fast heuristics (like FF and BF) and the more complex, theoretically-optimal (but often impractical) approximation schemes (APTAS).