3 Heuristic

3.1 Overview

Heuristic algorithms: Combine two algorithms, including Guillotine and Maximal Rectangles *[Jukka Jylänki, "A Thousand Ways to Pack the Bin", 2010]*.

Concept of free rectangles: A list of free rectangles represents the free space of the bin. In Guillotine algorithm, these rectangles are pairwise disjoint.

3.2 Sorting Input

3.2.1 Bins: Ascending order of density (cost/area), ties broken with the descending order of the longer side, followed by the descending order of the shorter side.

3.2.2 Items: Descending order of the longer side, ties broken with the descending order of the shorter side.

3.3 Choosing the Destination for Items:

3.3.1 Destination Bin: Bin First Fit rule – pack the item into the bin with the lowest index (after the process of sorting bins from 3.1.1); in other words, pack in the first bin that the item fits.

3.3.2 Destination Free Rectangle of a Bin: Best Short Side Fit rule – choose a free rectangle where the shorter remainder side after insertion is minimized; in other words, minimize the length of the shorter leftover side, ties broken with best longer side (longer leftover side is minimized).

3.4 Packing process:

3.4.1 Guillotine:
- Pack the item into the first free rectangle of the bin, which means the bin itself, starting with its bottom-left corner.
- For each insertion, split the initial free rectangle into smaller free rectangle(s) by the Guillotine split rule, then tracked in a list.
- Whenever a new item is inserted into the bin, choose a free rectangle (with the rule 3.2.2) and place the item into its bottom-left corner, then split the chosen rectangle using the Guillotine split rule to produce at most two new rectangles.
- Merge some free rectangles into larger ones if possible.

* **Splitting rule:** Best Short Side rule - split by horizontal axis if the free rectangle's width is less than its height; otherwise, split by vertical axis.
* **Rectangle merging:** If exists a pair of neighboring rectangles F_i and F_j such that the union $F_i \cup F_j$ can be exactly represented by a single bigger rectangle, merge these two into one.

3.4.2 Maximal Rectangles:
- Rather than choosing one of the two split axes like in the Guillotine algorithm, the Maximal Rectangles algorithm picks both split axes at the same time to ensure that the largest possible rectangular areas are present in the list of free rectangles.
- Because the free rectangles are no longer pairwise disjoint, any free rectangle that intersects the area occupied by the newly inserted item is split such to remove the intersection.
- Delete every free rectangle which is fully overlapped by others in the list.

Pseudo-code:

Algorithm 1: The Guillotine algorithm

Initialize:
Set $F = \{(W, H)\}$;
Pack:
foreach *Item i = (w, h) in the list of inserted items of the bin* **do**
  Decide the free rectangle $F_j \in F$ to pack the item into;
  Decide the orientation for the item and place it at the bottom-left of F_j;
  Use the guillotine split scheme to subdivide F_j into two new free rectangles F_{j_1} and F_{j_2};
  Set $F \leftarrow F \cup \{F_{j_1}, F_{j_2}\} \backslash \{F_j\}$;
  foreach *Ordered pair of free rectangles* F_{j_1}, F_{j_2} *in* F **do**
    if F_{j_1} *and* F_{j_2} *can be merge together* **then**
      F_{merge} = Merge F_{j_1} and F_{j_2};
      Set $F \leftarrow F \cup \{F_{merge}\} \backslash \{F_{j_1}, F_{j_2}\}$;
    end
  end
end

Algorithm 2: The Maximal Rectangles algorithm

Initialize:
Set $F = \{(W, H)\}$;
Pack:
foreach *Item i = (w, h) in the list of inserted items of the bin* **do**
  Decide the free rectangle $F_j \in F$ to pack the item into;
  Decide the orientation for the item and place it at the bottom-left of F_j;
  Use the max_rec split scheme to subdivide F_j into two new free rectangles F_{j_1} and F_{j_2};
  Set $F \leftarrow F \cup \{F_{j_1}, F_{j_2}\} \backslash \{F_j\}$;
  foreach *Free Rectangles* F_j *in* F **do**
    Compute $F_j \backslash i$ and subdivided the result into at most four new free rectangles $F_{j_1}, ..., F_{j_4}$;
    Set $F \leftarrow F \cup \{F_{j_1}, ..., F_{j_4}\} \backslash \{F_j\}$;
  end
  foreach *Ordered pair of free rectangles* F_{j_1}, F_{j_2} *in* F **do**
    if F_{j_1} *contains* F_{j_2} **then**
      Set $F \leftarrow F \backslash \{F_{j_2}\}$;
    end
  end
end