Algorithm 1 Knapsack padding

Input: Antenna number M, k candidate packets $packets$
Output: Maximized value Schedule of packet for every stream $streams$

$used\_packet \leftarrow \emptyset$
$longest\_packet \leftarrow packets[0]$
for each $packet$ in $packets$ **do**
  if $packet.length > longes\_packet.length$ **then**
    $longest\_packet \leftarrow packet$
  end if
end for
$master\_length \leftarrow longest\_packet.length$
$streams[0][master\_length].packets \leftarrow longest\_packet$
$streams[0][master\_length].max\_value \leftarrow longest\_packet.value$
for $i = 1 : M - 1$ **do**
  for $j = 0 : master\_length - 1$ **do**
    $streams[i][j].max\_value \leftarrow 0$
    $streams[i][j].packets \leftarrow \emptyset$
    for each $packet$ in $packets$ **do**
      if $packet.length < j$ and $packet$ does not belong to $stream[i][j - packet.length] \bigcup used\_packet$ **then**
        $value \leftarrow packet.value + stream[i][j - packet.length] -$ LOSS$(stream[i-1][master_length], j, packet)$
        if $value > stream[i][j].max\_value$ **then**
          $streams[i][j].max\_value \leftarrow value$
          $streams[i][j].packet \leftarrow streams[i][j - packet.lenght].packets.append(packet)$
        end if
      end if
    end for
  end for
  $used.add(stream[i][master\_lenght].packets)$
  output $streams[i][master\_length]$
end for