Algorithm 2 joinSnapshot(R-tree R_P for indoor POIs, A1R-tree R_O for OTT, time t, integer k)

initialize an in-memory aggregate R-tree R_I
LeafEntrySet $les \leftarrow R_O$.PointQuery(t)
for each leaf entry $le \in les$ **do**
  $o \leftarrow le.Ptr_c.objectID$
  if $le.Ptr_p.t_e < t < le.Ptr_c.t_s$ **then** ▷ Inactive state
    $mbr_1 \leftarrow$ extend MBR($le.Ptr_p.deviceID.Range$) by $V_{max} \cdot (t - le.Ptr_p.t_e)$
    $mbr_2 \leftarrow$ extend MBR($le.Ptr_c.deviceID.Range$) by $V_{max} \cdot (le.Ptr_c.t_s - t)$
    $mbr \leftarrow$ MBR(mbr_1, mbr_2)
  else
    $mbr \leftarrow$ MBR($le.Ptr_c.deviceID.Range$)
  insert (o, mbr) into R_I
initialize a priority queue Q
for each entry e_P in $R_P.root$ **do**
  $ubFlow \leftarrow 0$; $list \leftarrow \varnothing$
  for each entry e_I in $R_I.root$ **do**
    if $e_P.mbr$ intersects $e_I.mbr$ **then**
      $ubFlow \leftarrow ubFlow + e_I.count$; $list \leftarrow list \cup \{e_I\}$
  $Q.enqueue(\langle e_P, list, ubFlow\rangle)$
$result \leftarrow \varnothing$; initialize a hash table H_U
while Q is not empty **do**
  $\langle e_P, list\rangle \leftarrow Q.dequeue()$
  if e_P is a leaf entry **then**
    if $list$ is null **then**
      add POI $e_P.object$ to $result$
      if $result = k$ **then return** $result$
    else
      if $list$ contain leaf entries **then**
        $flow \leftarrow 0$
        for each entry $e_I \in list$ **do**
          if $H_U[e_I.object] = \varnothing$ **then**
            $H_U[e_I.object] \leftarrow UR(e_I.object, t)$
          $flow \leftarrow flow + \phi_{t,e_P.object}(e_I.object)$
        if $flow \neq 0$ **then** $Q.enqueue(\langle e_P, null, flow\rangle)$
      else
        expandList($e_P, list$)
  else
    if $list$ contain leaf entries **then**
      for each sub-entry e'_P in $e_P.node$ **do**
        $ubFlow \leftarrow 0$; $list2 \leftarrow \varnothing$
        for each entry $e'_I \in list$ **do**
          if $e'_P.mbr$ intersects $e'_I.mbr$ **then**
            $ubFlow \leftarrow ubFlow + 1$
            $list2 \leftarrow list2 \cup \{e'_I\}$
        if $list2 \neq \varnothing$ **then**
          $Q.enqueue(\langle e'_P, list2, ubFlow\rangle)$
    else
      for each sub-entry e'_P in $e_P.node$ **do**
        expandList($e'_P, list$)