# Heuristic Algorithms - WISE

**Algorithm 1:** Candidates Selection Heuristic

**input** : Complete Network Graph: $G(S, L)$
Coverage Threshold: $Th_{cov}$
Current Attestation Time: $T_{current}$
Updated Devices' Hidden Markov Models : $\{\theta_1, \ldots, \theta_{|S|}\}$

**output** : Candidate Nodes: $N_{attest}$

**begin**
  //Initialize the set of candidate nodes.
  $N_{attest} \leftarrow \{\}$;
  //Initialize paired $\langle$Node, Attestation Probability$\rangle$ Set.
  $N_{prob} \leftarrow \{\}$;
  //Loop over each node $D_i$ in $S$.
  **foreach** $D_i \in S$ **do**
    //Compute and add the attestation probability of node $D_i$.
    $N_{prob} \leftarrow N_{prob} \cup \{\langle n_i, AttesationProbability(D_i, \theta_i)\rangle\}$;
  //Sort nodes ascending based on their attestation probability.
  $N_{sorted} \leftarrow SortNodes(N_{prob})$;
  //Loop over sorted nodes and pick up the most likely compromised ones.
  **foreach** $\langle D_i, \_\rangle \in N_{sorted}$ **do**
    //add the current node to candidate subset of nodes.
    $N_{attest} \leftarrow N_{attest} \cup \{D_i\}$
    //Check if the candidate subset of nodes satisfies the coverage ratio.
    **if** $\frac{|N_{attest}|}{|S|} \geq Th_{cov}$ **then**
      $Break$;
  //Loop over each node $D_i$ in $S$.
  **foreach** $D_i \in S$ **do**
    //Check and add attestation time-based violated device.
    **if** $D_i.T_{last} + D_i.T_{max} - T_{current} \leq 0$ **then**
      $N_{attest} \leftarrow N_{attest} \cup \{D_i\}$
  **return** $N_{attest}$

**Algorithm 2:** Gap Bridging Heuristic

**input** : Candidate Nodes: $N_{attest}$

**output** : Bridges (set of nodes): $N_{bridge}$

**begin**
  //Initialize the set of intermediate nodes.
  $N_{bridge} \leftarrow \{\}$;
  //Loop over each node in $N_{attest}$
  **foreach** $D_i \in N_{attest}$ **do**
    //Temporary set to hold a 2-tuple element containing the path of the node and its criticality degree.
    $P_{temp} \leftarrow \{\}$;
    //Loop over top X shortest paths of node $D_i$.
    **foreach** $p \in D_i.D_{paths}$ **do**
      //Compute and store path's criticality.
      $P_{temp} \leftarrow P_{temp} \cup \langle p, \sum_{D_i \in p} D_i.D_{degree}\rangle$;
    //Get the path that has minimum nodes' degree criticality.
    $\langle p_{min}, \_\rangle \leftarrow Min(P_{temp})$;
    //Add the nodes of minimum critical path to the set of intermediate nodes.
    $N_{bridge} \leftarrow N_{bridge} \cup p_{min}$;
  **return** $N_{bridge}$

**Algorithm 3:** Cluster Selection Heuristic.

**input** : Candidate nodes: $N_{attest}$
Intermediate Nodes: $N_{bridge}$
Clusters sets (Categories): $\{C_1, C_2, \ldots, C_k\}$

**output** : Attestation Clusters: $C_{attest}$
Intermediate Clusters: $C_{bridge}$

**begin**
  //Initialize attestation and bridging clusters sets.
  $C_{attest} \leftarrow \{\}$;
  $C_{bridge} \leftarrow \{\}$;
  //Initialize a set for keeping a 2-tuple element of a cluster with its communication overhead in terms of number of hops.
  $C_{cost} \leftarrow \{\}$;
  //Loop over k categories.
  **foreach** $C_j \in \{C_1, C_2, \ldots, C_K\}$ **do**
    //Loop over clusters in the $j^{th}$ category.
    **foreach** $c_{ji} \in C_j$ **do**
      //Check if $c_{ji}$ cluster covers at least one node of either $N_{attest}$ or $N_{bridge}$. If so, compute and store communication overhead of $c_{ji}$.
      **if** $c_{ji} \cap (N_{attest} \cup N_{bridge}) \neq \emptyset$ **then**
        $C_{cost} \leftarrow C_{cost} \cup \{\langle c_{ji}, ComOverhead(c_{ji})\rangle\}$;
  //Sort selected clusters in an ascending way.
  $C_{sorted\_cost} \leftarrow SortComOverhead(C_{cost})$
  //Loop over sorted clusters.
  **foreach** $\langle c_{ij}, \_\rangle \in C_{sorted\_cost}$ **do**
    //Check if the cluster $c_{ji}$ covers node in $N_{attest}$.
    **if** $c_{ji} \cap N_{attest} \neq \emptyset$ **then**
      //Add the cluster to the attestation-targeted set of clusters.
      $C_{attest} \leftarrow C_{attest} \cup c_{ji}$
      //Remove the covered node(s) by the cluster $c_{ji}$ from $N_{attest}$ set and $N_{bridge}$ set if they exist.
      $N_{attest} \leftarrow N_{attest} - c_{ji} \cap N_{attest}$
      $N_{bridge} \leftarrow N_{bridge} - c_{ji} \cap N_{attest}$
    //Check if the cluster $c_{ji}$ covers any node in $N_{bridge}$.
    **else if** $c_{ji} \cap N_{bridge} \neq \emptyset$ **then**
      //Add the cluster to the bridge-targeted set of clusters.
      $C_{bridge} \leftarrow C_{bridge} \cup c_{ji}$
      //Remove the covered node(s) by the cluster $c_{ji}$ from $N_{bridge}$ set if exists.
      $N_{bridge} \leftarrow N_{bridge} - c_{ji} \cap N_{bridge}$
  **return** $C_{attest}, C_{bridge}$