Appendix I

Algorithm 1 A graph privacy-preserving algorithm achieving $k$-anonymity by adding edges. Based on the idea of $k$-anonymity, we define the concept of the equivalence class of structural features. The equivalence class of structural feature, such as degree and hub fingerprint, refers to a combination of elements with the same feature. The elements can be nodes, node pairs and so on.

Input: A set comprising the equivalence classes of structural features: $S$; A set comprising the $\Delta k$ of equivalence classes: $\Delta S$; $\Delta k$ is the difference between the target number of elements in the equivalence class and the actual number of elements in the equivalence class.

Output: A set of edges to be added: $edgeAddList$;

1: function $GRAPHPRIVACY\_PRESERVE(S, \Delta S)$
2:totalNodeInfo = []
3: for $s$ in $S$ do
4: $(pullCost, pullNodeInfo) = PULL(S, \Delta S, s)$ $\triangleright$ Make the current equivalence class satisfy $k$-anonymity by adding edges to the elements in other equivalence classes and transferring them into $s$
5: $(pushCost, pushNodeInfo) = PUSH(S, \Delta S, s)$ $\triangleright$ Make the number of elements in the current equivalence class to be 0, by adding edges to the elements in $s$ and transferring them into other equivalence classes
6: end for
7: if $pullCost = pushCost = \infty$ then
8: return None
9: end if
10: if $pullCost <= pushCost$ then
11: totalNodeInfo += pullNodeInfo
12: else
13: totalNodeInfo += pushNodeInfo
14: end if
15: $edgeAddList = CONVERT(totalNodeInfo)$
16: return $edgeAddList$
17: end function

18: function $PULL(S, \Delta S, s)$
19: $\Delta k_c = \Delta S[s]$, pullCount = 0, pullCost = 0, pullNodeInfo = []
20: pullSet = GetPullSet($s$) $\triangleright$ Get a set of equivalence classes that can transfer elements to $s$
21: for $s_j$ in pullSet do
22: $\Delta k_j = \Delta S[s_j]$
23: if $\Delta k_j < 0$ then
24: $(edgeCost, priorityCost, nodeInfo) = TRANSFER(s_j, s, isLocked = false)$ $\triangleright$ Transfer adequate unlocked idle elements in the $s_j$ to $s$ and calculate the number of added edges, the cost of priority of nodes and the node information that need to be added to the current solution
25: pullCost += Combine(edgeCost, priorityCost)
26: pullNodeInfo += nodeInfo
27: if size(nodeInfo) <= $\Delta k_c - pullCount$ then
28: pullCount += size(nodeInfo)
29: else
30: pullCount = $\Delta k_c$
31: break
32: end if
33: end if
34: end for
35: if pullCount! = $\Delta k_c$ then
36:
37:      pullCost = ∞
38:   end if
39:   return (pullCost, pullNodeInfo)
40: end function

42: function PUSH(S, ∆S, s)
43:   Δk_c = ∆S[s], pushCount = 0, pushCost = 0, pushNodeInfo = []
44:   pushSet = GetPushSet(s) ▷ Get a set of equivalence classes that can transfer elements of s to
45:   for s_j in pushSet do
46:     Δk_j = ∆S[s_j]
47:     if Δk_j > 0 then
48:       if Δk_j >= pushCount then
49:         transferCount = pushCount
50:       else
51:         transferCount = Δj
52:     end if
53:     [edgeCost, priorityCost, nodeInfo] = Transfer(s, s_j, transferCount) ▷ Transfer transferCount
54:       elements in the current equivalence class to s_j
55:     pushCost += Combine(edgeCost, priorityCost)
56:     pushNodeInfo += nodeInfo
57:     pushCount -= transferCount
58:     if pushCount == 0 then
59:       break
60:     end if
61:   end for
62:   if pushCount! = 0 then
63:     pushCost = ∞
64:   end if
65:   return (pushCost, pushNodeInfo)
66: end function