
Algorithm 1 SpeakEasy2

```
1: function SPEAKEASY2(Adjacency  $A$ )
2:    $R \leftarrow 10$  ▷ Set number of runs to perform
3:    $P \leftarrow \text{LIST}()$  ▷ CONTAINER FOR POSSIBLE PARTITIONS
4:   for  $i \leftarrow 1$  to  $R$  do
5:      $p \leftarrow \text{LABELNODES}(A)$ 
6:      $P.\text{append}(p)$ 
7:   end for
8:    $P.\text{flatten}()$ 
9:    $p \leftarrow \text{SELECTREPRESENTATIVEPARTITION}(P)$ 
10:  return  $p$ 
11: end function

12: function LABELNODES(ADJACENCY  $A$ )
13:    $P \leftarrow \text{LIST}()$ 
14:    $p \leftarrow \text{INITIALLABELS}(A)$ 
15:    $\text{merge-p} \leftarrow \text{false}$ 
16:    $n_{\text{bubbles}} \leftarrow 0$  ▷ NUMBER OF TIMES BUBBLE STAGE RUN
17:    $n_{\text{merged}} \leftarrow 0$  ▷ NUMBER OF MERGED COMMUNITY DURING LAST MERGE PHASE
18:    $n_{\text{partitions}} \leftarrow 5$  ▷ SET NUMBER OF PARTITIONS TO FIND
19:    $t \leftarrow 1$ 
20:   while  $\text{length}(P) < n_{\text{partitions}}$  do
21:      $M \leftarrow \text{SELECTMODE}(t, \text{merge-p})$ 
22:     if  $M$  IS TYPICAL then
23:        $p \leftarrow \text{TYPICAL}(A, p)$ 
24:     else if  $M$  IS NURTURE then
25:        $p \leftarrow \text{NURTURE}(A, p)$ 
26:     else if  $M$  IS BUBBLE then
27:        $p \leftarrow \text{BUBBLE}(A, p)$ 
28:        $n_{\text{bubbles}} \leftarrow n_{\text{bubbles}} + 1$ 
29:       if  $n_{\text{bubbles}} > 2$  then
30:          $\text{merge-p} \leftarrow \text{true}$  ▷ CONSIDER MERGING ONLY AFTER A FEW INITIAL BUBBLE PHASES
31:       end if
32:     else if  $M$  IS MERGE then
33:        $n_{\text{bubbles}} \leftarrow 0$ 
34:        $p, n_{\text{merged}} \leftarrow \text{MERGE}(A, p)$ 
35:       if  $n_{\text{merged}} = 0$  then ▷ ONLY ADD PARTITION IF STABLE
36:          $P.\text{append}(p)$ 
37:          $\text{merge-p} \leftarrow \text{false}$  ▷ TURN OFF MERGING UNTIL PARTITION HAS BEEN BUBBLED
38:       end if
39:     end if
40:      $t \leftarrow t + 1$ 
41:   end while
42:   return  $P$ 
43: end function

44: function INITIALLABELS(ADJACENCY  $A$ )
45:    $P \leftarrow \text{VECTOR}(|V(A)|)$  ▷ CONTAINER FOR RANDOM LABELS
46:    $n_{\text{labels}} \leftarrow \text{ceil}\left(\frac{|V(A)|}{100}\right)$ 
47:   for  $\nu \in V(A)$  do
48:     if  $|\{v \mid (v, \nu) \in E(A), v \neq \nu\}| > 0$  then ▷ CHECK IF NODE HAS INCOMING CONNECTIONS
49:        $P[\nu] \leftarrow \text{randint}(1, n_{\text{labels}})$ 
50:     end if
51:   end for
52:   for  $\nu \in V(A)$  do
53:      $l_{\text{max}} \leftarrow \max(P)$ 
```

```

54:     if  $|\{v \mid (v, \nu) \in E(A), v \neq \nu\}| = 0$  then
55:          $P[\nu] \leftarrow l_{\max} + 1$ 
56:     end if
57: end for
58: return  $P$ 
59: end function

60: function SELECTMODE(TIMESTEP  $t$ , PREDICATE merge-p)
61:     if  $t \leq 20$  then
62:         return TYPICAL
63:     end if
64:     if merge-p then
65:         if TIMESINCELASTMERGE  $> 1$  and TIMESINCELASTBUBBLE  $> 3$  then
66:             return MERGE
67:         end if
68:     else
69:         if TIMESINCELASTMERGE  $> 2$  and TIMESINCELASTBUBBLE  $> 14$  then
70:             return BUBBLE
71:         end if
72:         if TIMESINCELASTMERGE  $> 1$  and TIMESINCELASTBUBBLE  $< 5$  then
73:             return NURTURE
74:         end if
75:     end if
76:     return TYPICAL
77: end function

78: function MOSTSPECIFICLABEL(NODE  $\nu$ , ADJACENCY  $A$ , PARTITION  $P$ )
79:      $l' \leftarrow 0$ 
80:      $q \leftarrow -\infty$ 
81:     for  $l \in \{P\}$  do
82:          $o \leftarrow \sum_{i \in A, i \neq \nu} A[i, \nu] \delta(P[i], l)$ 
83:          $e \leftarrow \left( \frac{\sum_{i, j \in A, i \neq j} A[i, j] \delta(P[i], l)}{\sum_{i, j \in A, i \neq j} A[i, j]} \right) \sum_{i \in A, i \neq \nu} A[i, \nu]$ 
84:         if  $o - e > q$  then
85:              $l' \leftarrow l$ 
86:              $q \leftarrow o - e$ 
87:         end if
88:     end for
89:     return  $l', q$ 
90: end function

91: function LOWESTSPECIFICITYLABELS(ADJACENCY  $A$ , PARTITION  $P$ , THRESHOLD  $\theta$ )
92:      $n_{\text{update}} \leftarrow \text{ceil}(\theta \times |V(A)|)$ 
93:      $Q \leftarrow \text{VECTOR}(V(A))$ 
94:      $N \leftarrow \text{VECTOR}(n_{\text{update}})$ 
95:     for  $\nu \in V(A)$  do
96:          $Q[\nu] \leftarrow \text{MOSTSPECIFICLABEL}(\nu, A, P)$ 
97:     end for
98:     for  $i \leftarrow 0$  to  $i < n_{\text{update}}$  do
99:          $N[i] = Q.\text{pop}(\arg \min_{\nu}(Q))$ 
100:     end for
101:     return  $N$ 
102: end function

103: function TYPICAL(ADJACENCY  $A$ , PARTITION  $P$ )
104:      $P_{\text{updated}} \leftarrow P$ 
105:     for  $\nu \in (V' \subset V(A))$  do

```

▷ CHECK IF NODE HAS NO INCOMING CONNECTIONS

▷ HOLDS CURRENT BEST LABEL

▷ HOLDS CURRENT BEST LABEL'S SPECIFICITY

▷ OBSERVED SUM OF NEIGHBORS LABELED l

▷ EXPECTED SUM OF NEIGHBORS LABELED l

▷ DEFAULT TO TYPICAL MODE

▷ ONLY UPDATE A SUBSET OF ALL NODES (90%)

```

106:    $P_{\text{updated}}[\nu, -] \leftarrow \text{MOSTSPECIFICLABEL}(\nu, A, P)$ 
107: end for
108: return  $P_{\text{updated}}$ 
109: end function

110: function NURTURE(ADJACENCY  $A$ , PARTITION  $P$ )
111:    $P_{\text{updated}} \leftarrow P$ 
112:    $N \leftarrow \text{LOWESTSPECIFICITYLABELS}(A, P, 0.9)$  ▷ WORST 90% OF NODE LABELS
113:   for  $\nu \in N$  do ▷ ONLY UPDATE POORLY LABELED NODES
114:      $P_{\text{updated}}[\nu, -] \leftarrow \text{MOSTSPECIFICLABEL}(\nu, A, P)$ 
115:   end for
116:   return  $P_{\text{updated}}$ 
117: end function

118: function BUBBLE(ADJACENCY  $A$ , PARTITION  $P$ )
119:    $L \leftarrow \{\{\nu \mid \nu \in V(A), P[\nu] = l\} \mid l \in \{P\}\}$  ▷ SETS OF NODES IN EACH LABEL
120:    $P_{\text{updated}} \leftarrow P$ 
121:    $N \leftarrow \text{LOWESTSPECIFICITYLABELS}(A, P, 0.9)$  ▷ WORST 90% OF NODE LABELS
122:    $\tilde{n} \leftarrow \text{median}(\{|l| \mid l \in L\})$  ▷ MEDIAN COMMUNITY SIZE
123:   for  $l \in L$  do ▷ SET OF NODES IN LABEL
124:      $l_{\text{max}} \leftarrow \max(P_{\text{updated}})$ 
125:     if  $|l| > 4$  then ▷ DON'T BURST SMALL COMMUNITIES
126:       for  $\nu \in (l \cap N)$  do ▷ LOW QUALITY NODES IN THE COMMUNITY
127:          $P_{\text{updated}}[\nu] \leftarrow \text{randint}\left(1, \frac{|(l \cap N)|}{\tilde{n}}\right) + l_{\text{max}}$  ▷ RANDOMLY ASSIGN TO NEW COMMUNITIES12
128:       end for
129:     end if
130:   end for
131:   return  $P_{\text{updated}}$ 
132: end function

133: function MERGE(ADJACENCY  $A$ , PARTITION  $P$ )
134:    $L \leftarrow \text{VECTOR}(|P|)$  ▷ CONTAINER FOR LABEL THAT SENDS MOST CROSSTALK TO  $i_{th}$  LABEL
135:    $C \leftarrow \text{VECTOR}(|P|)$  ▷ CONTAINER FOR AMOUNT OF CROSSTALK SENT BY LABEL IN  $L$ 
136:    $P_{\text{updated}} \leftarrow P$ 
137:    $n_{\text{merged}} \leftarrow 0$ 
138:   for  $l \in \{P\}$  do
139:      $L[l], C[l] \leftarrow \text{MAXCROSSTALK}(A, P, l)$ 
140:   end for
141:    $\bar{c} \leftarrow \text{mean}(C)$ 
142:   for  $l \in \{P\}$  do
143:     if  $C[l] > \bar{c}, L[l] \in \{P_{\text{updated}}\}$  then ▷ ENSURE  $L[l]$  HASN'T ALREADY BEEN MERGED INTO  $l$ 
144:       for  $\nu \in V(A)$  do
145:         if  $P[\nu] = l$  then
146:            $P_{\text{updated}}[l] \leftarrow L[l]$ 
147:         end if
148:       end for
149:        $n_{\text{merged}} \leftarrow n_{\text{merged}} + 1$ 
150:     end if
151:   end for
152:   return  $P_{\text{updated}}, n_{\text{merged}}$ 
153: end function ▷ HOW MUCH NODES IN ONE LABEL HEARD FROM NODES IN ANOTHER LABEL

154: function MAXCROSSTALK(ADJACENCY  $A$ , PARTITION  $P$ , LABEL  $l$ )
155:    $l' \leftarrow 0$  ▷ HOLDS CURRENT HIGHEST CROSSTALK LABEL

```

¹New communities should be approximately the size of the median community size.

²New communities are split from old communities; nodes can't end up in a community with a node that was previously in a different community.

```

156:  $c \leftarrow -\infty$  ▷ HOLDS CURRENT HIGHEST AMOUNT OF CROSSTALK
157: for  $l_i \in \{P\}, l_i \neq l$  do
158:    $o \leftarrow \sum_{i,j \in A, i \neq j} A[i,j] \delta(P[i], l_i) \delta(P[j], l)$  ▷ SUM OF NODES IN  $l_i$  THAT TALK TO A NODE IN  $l$ 
159:    $e \leftarrow \left( \frac{\sum_{i,j \in A, P[i] \neq P[j]} A[i,j] \delta(P[i], l_i)}{\sum_{i,j \in A, P[i] \neq P[j]} A[i,j]} \right) \sum_{i,j \in A, P[i] \neq P[j]} A[i,j] \delta(P[j], l)$  ▷ EXPECTED TALK FROM  $l_i$  INTO  $l$ 
160:   if  $o - e > q$  then
161:      $l' \leftarrow l$ 
162:      $q \leftarrow o - e$ 
163:   end if
164: end for
165: return  $l', c$ 
166: end function

167: function SELECTREPRESENTATIVEPARTITION(PARTITIONS  $P$ )
168:    $p_{\text{best}} \leftarrow 0$  ▷ MOST REPRESENTATIVE PARTITION
169:    $q_{\text{best}} \leftarrow 0$  ▷ SUM OF MOST REPRESENTATIVE PARTITION'S NMIs
170:   for  $p_i \in P$  do
171:      $q \leftarrow \sum_{p_j \in P, p_j \neq p_i} \text{NMI}(p_i, p_j)$ 
172:     if  $q > q_{\text{best}}$  then
173:        $p_{\text{best}} \leftarrow p_i$ 
174:        $q_{\text{best}} \leftarrow q$ 
175:     end if
176:   end for
177:   return  $p_{\text{best}}$ 
178: end function

```
