Algorithm 1 SeedSubTrees(Γ)

1: **for** $comp \in \{Components\}$ **do**
2: $level = \{\}$; {A Level of a SubTree}
3: $nFailed = \{0, 0, ...0\}$; {Counts Failed Components}
4: $\beta = \{$Array of Linked Lists$\}$; {Breadth First History of Trees}
5: $\Gamma Cache = \{$Array of size $2^{|\Gamma_{comp}|}\}$; {Stores all possible subsets of Γ_{comp}}
6: **if** (NotEmpty(Γ_{comp})) **then**
7: append $comp$ to $level$;
8: $nFailed[comp] = 1$;
9: $\Gamma Cache_{comp} = \mathcal{P}(\Gamma_{comp})$; {Power Set (Ordered Set) is cached}
10: append | to $\beta[comp]$;
11: GrowSubTrees($level, nFailed, \beta, 1$);
12: **end if**
13: **end for**

Algorithm 2 GrowSubTrees(l, S, β, r)

1: $F = \text{failed}(L)$;
2: $\rho = \prod_{i=1}^{|F|} \Gamma_{F[i]}$; {Cartesian Product}
3: **for** $i \in \rho$ **do**
4: $nl = \{\}$; {Next level of SubTree to be built}
5: **for** $j \in \rho_i$ **do**
6: $p = L[i]$; {To be parent}
7: append $\Gamma Cache_p[j]$ to nl;
8: **for** $k \in \Gamma Cache_p[j]$ **do**
9: **if** MarkedToBeAdded(k) **then**
10: $S[k] = S[k] + 1$;
11: push k into β_k; {Store added child's information}
12: $r = r * \phi_{i,k}$; {Update SubTree rate}
13: **else**
14: push i into β_k; {Store parent info where child not added}
15: **end if**
16: **end for**
17: **end for**
18: **for** $s \in S$ **do**
19: **if** $S[s] >$redundancy(s) **then**
20: goto line 3; {SubTree is invalid because it has more components than in the system}
21: **end if**
22: **end for**
23: **if** AtleastOneChildAdded **then**
24: GrowSubTrees(nl, S, β, r); {Grow the SubTree further}
25: **else**
26: ProcessRates(l, S, β, r); {A stunted SubTree's rate is calculated and then it is discarded}
27: **end if**
28: **end for**